Description

A class containing methods to assist with accessibility for users with different vision capabilities.

Static Methods

GetColorBlindSafePalette	Gets a palette of colors that should be distinguishable for normal vision, deuteranopia, protanopia, and tritanopia.

Description

Unity Ads.

Static Properties

debugMode	Controls the amount of logging output from the advertisement system.
gameId	Returns the game identifier for the current platform.
isInitialized	Returns whether the advertisement system is initialized successfully.
isShowing	Returns whether an advertisement is currently being shown.
isSupported	Returns if the current platform is supported by the advertisement system.
testMode	Returns whether the testing mode is enabled.
version	Returns the current Unity Ads version.

Static Methods

GetPlacementState	Returns the placement state.
Initialize	Manually initializes the advertisement system. Normally this is done from editor, and you should only call this method if you are using UnityAds with automatic initialization disabled.
IsReady	Returns whether an advertisement is ready to be shown. Placements are configured per game in the UnityAds admin site, where you can also set your default placement.
SetMetaData	Sets various metadata for Unity Ads.
Show	Show an advertisement in your project.

Obsolete

Description

Player debug message level.

Description

Class for sending various metadata to UnityAds.

Properties

category	Metadata category.
Values	

Public Methods

Get	
Set	Sets new metadata fields.

Description

Collection of options that can be passed to Advertisements.Show to modify advertisement behaviour.

Properties

gamerSid	Add a string to specify an identifier for a specific user in the game.
resultCallback	Callback to recieve the result of the advertisement.

Description

Various states that Unity Ads placements can be in.

Properties

Ready	Placement is ready to be shown an advertisement from.
NotAvailable	Placement is not available.
Disabled	Placement has been disabled.
Waiting	Placement is waiting to be ready.
NoFill	Placement has no advertisements to show.

Description

ShowResult is passed to ShowOptions.resultCallback after the advertisement has completed.

Properties

Failed	Indicates that the advertisement failed to complete.
Skipped	Indicates that the advertisement was skipped.
Finished	Indicates that the advertisement completed successfully.

Description

Singleton class to access the baked NavMesh.

The NavMesh is a class can be used to do spatial queries, like pathfinding and walkability tests, set the pathfinding cost for specific area types, and to tweak global behavior of pathfinding and avoidance.

In order to use the spatial queries, you need to first bake NavMesh for your Scene.

See also:
- Building a NavMesh – for more information on how to setup and bake NavMesh
- Areas and Costs – to learn how to use different Area types.
- NavMeshAgent – to learn how to control and move NavMesh Agents.
- NavMeshObstacle – to learn how to control NavMesh Obstacles using scripting.
- OffMeshLink – to learn how to control Off-Mesh Links using scripting.

Static Properties

AllAreas	Area mask constant that includes all NavMesh areas.
avoidancePredictionTime	Describes how far in the future the agents predict collisions for avoidance.
onPreUpdate	Set a function to be called before the NavMesh is updated during the frame update execution.
pathfindingIterationsPerFrame	The maximum amount of nodes processed each frame in the asynchronous pathfinding process.

Static Methods

AddLink	Adds a link to the NavMesh. The link is described by the NavMeshLinkData struct.
AddNavMeshData	Adds the specified NavMeshData to the game.
CalculatePath	Calculate a path between two points and store the resulting path.
CalculateTriangulation	Calculates triangulation of the current navmesh.
CreateSettings	Creates and returns a new entry of NavMesh build settings available for runtime NavMesh building.
FindClosestEdge	Locate the closest NavMesh edge from a point on the NavMesh.
GetAreaCost	Gets the cost for path finding over geometry of the area type.
GetAreaFromName	Returns the area index for a named NavMesh area type.

GetSettingsByID	Returns an existing entry of NavMesh build settings.
GetSettingsByIndex	Returns an existing entry of NavMesh build settings by its ordered index.
GetSettingsCount	Returns the number of registered NavMesh build settings.
GetSettingsNameFromID	Returns the name associated with the NavMesh build settings matching the provided agent type ID.
Raycast	Trace a line between two points on the NavMesh.
RemoveAllNavMeshData	Removes all NavMesh surfaces and links from the game.
RemoveLink	Removes a link from the NavMesh.
RemoveNavMeshData	Removes the specified NavMeshDataInstance from the game, making it unavailable for agents and queries.
RemoveSettings	Removes the build settings matching the agent type ID.
SamplePosition	Finds the closest point on NavMesh within specified range.
SetAreaCost	Sets the cost for finding path over geometry of the area type on all agents.

Delegates

OnNavMeshPreUpdate	A delegate which can be used to register callback methods to be invoked before the NavMesh system updates.

Description

Navigation mesh agent.

This component is attached to a mobile character in the game to allow it to navigate the Scene using the NavMesh. See the Navigation section of the manual for further details.

Properties

acceleration	The maximum acceleration of an agent as it follows a path, given in units / sec^2.
agentTypeID	The type ID for the agent.
angularSpeed	Maximum turning speed in (deg/s) while following a path.
areaMask	Specifies which NavMesh areas are passable. Changing areaMask will make the path stale (see isPathStale).
autoBraking	Should the agent brake automatically to avoid overshooting the destination point?
autoRepath	Should the agent attempt to acquire a new path if the existing path becomes invalid?
autoTraverseOffMeshLink	Should the agent move across OffMeshLinks automatically?
avoidancePriority	The avoidance priority level.
baseOffset	The relative vertical displacement of the owning GameObject.
currentOffMeshLinkData	The current OffMeshLinkData.
desiredVelocity	The desired velocity of the agent including any potential contribution from avoidance. (Read Only)
destination	Gets or attempts to set the destination of the agent in world-space units.
hasPath	Does the agent currently have a path? (Read Only)
height	The height of the agent for purposes of passing under obstacles, etc.
isOnNavMesh	Is the agent currently bound to the navmesh? (Read Only)
isOnOffMeshLink	Is the agent currently positioned on an OffMeshLink? (Read Only)
isPathStale	Is the current path stale. (Read Only)
isStopped	This property holds the stop or resume condition of the NavMesh agent.
navMeshOwner	Returns the owning object of the NavMesh the agent is currently placed on (Read

	Only).
nextOffMeshLinkData	The next OffMeshLinkData on the current path.
nextPosition	Gets or sets the simulation position of the navmesh agent.
obstacleAvoidanceType	The level of quality of avoidance.
path	Property to get and set the current path.
pathPending	Is a path in the process of being computed but not yet ready? (Read Only)
pathStatus	The status of the current path (complete, partial or invalid).
radius	The avoidance radius for the agent.
remainingDistance	The distance between the agent's position and the destination on the current path. (Read Only)
speed	Maximum movement speed when following a path.
steeringTarget	Get the current steering target along the path. (Read Only)
stoppingDistance	Stop within this distance from the target position.
updatePosition	Gets or sets whether the transform position is synchronized with the simulated agent position. The default value is true.
updateRotation	Should the agent update the transform orientation?
updateUpAxis	Allows you to specify whether the agent should be aligned to the up-axis of the NavMesh or link that it is placed on.
velocity	Access the current velocity of the NavMeshAgent component, or set a velocity to control the agent manually.

Public Methods

ActivateCurrentOffMeshLink	Enables or disables the current off-mesh link.
CalculatePath	Calculate a path to a specified point and store the resulting path.
CompleteOffMeshLink	Completes the movement on the current OffMeshLink.
FindClosestEdge	Locate the closest NavMesh edge.
GetAreaCost	Gets the cost for path calculation when crossing area of a particular type.
Move	Apply relative movement to current position.
Raycast	Trace a straight path towards a target postion in the NavMesh without moving the agent.

ResetPath	Clears the current path.
SamplePathPosition	Sample a position along the current path.
SetAreaCost	Sets the cost for traversing over areas of the area type.
SetDestination	Sets or updates the destination thus triggering the calculation for a new path.
SetPath	Assign a new path to this agent.
Warp	Warps agent to the provided position.

Inherited Members

Properties

enabled	Enabled Behaviours are Updated, disabled Behaviours are not.
isActiveAndEnabled	Has the Behaviour had active and enabled called?
gameObject	The game object this component is attached to. A component is always attached to a game object.
tag	The tag of this game object.
transform	The Transform attached to this GameObject.
hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.

Public Methods

BroadcastMessage	Calls the method named methodName on every MonoBehaviour in this game object or any of its children.
CompareTag	Is this game object tagged with tag ?
GetComponent	Returns the component of Type type if the game object has one attached, null if it doesn't.
GetComponentInChildren	Returns the component of Type type in the GameObject or any of its children using depth first search.
GetComponentInParent	Returns the component of Type type in the GameObject or any of its parents.
GetComponents	Returns all components of Type type in the GameObject.
GetComponentsInChildren	Returns all components of Type type in the GameObject or any of its children.
GetComponentsInParent	Returns all components of Type type in the GameObject or any of its parents.

SendMessage	Calls the method named methodName on every MonoBehaviour in this game object.
SendMessageUpwards	Calls the method named methodName on every MonoBehaviour in this game object and on every ancestor of the behaviour.
GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.

Static Methods

Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.

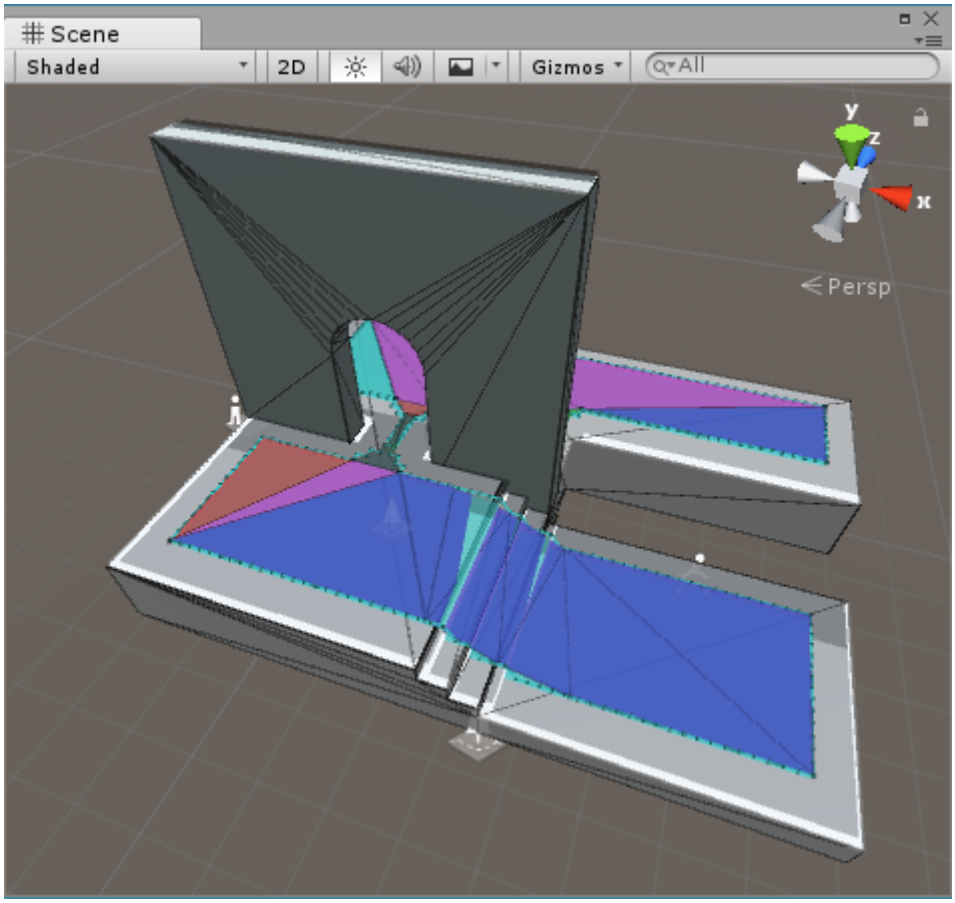
Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Description

Specify which of the temporary data generated while building the NavMesh should be retained in memory after the process has completed.

It is possible to collect and display in the Editor the intermediate data used in the process of building the navigation mesh using the NavMeshBuilder. This can help with diagnosing those situations when the resulting NavMesh isn't of the expected shape.



Input Geometry, Regions, Polygonal Mesh Detail and Raw Contours shown after building the NavMesh with debug options

The process for computing a NavMesh comprises of several sequential steps:
i. decomposing the Scene's terrain and meshes into triangles;
ii. rasterizing the input triangles into a 3D voxel representation and finding ledges;
iii. partitioning the voxels lying at the surface into simpler horizontal regions;
iv. finding a tight-fitting contour for each of these regions;
v. simplifying the contours into polygonal shapes;
vi. creating a mesh of convex polygons based on all the contours combined;
vii. refining the polygonal mesh into a triangulated version that approximates better the Scene's original geometry.

Through the use of the debug functionality the results from each stage can be captured and displayed separately, whereas normally they would get discarded when the NavMesh construction is completed.

Depending on the Scene composition this debug data can be considerably large in size. It is stored in memory in a compressed manner but gets further expanded when being displayed.

Notes:
1. Unity does not save Debug visualizations - they are only available during the session in which Unity is building the NavMesh.
2. Debug data is neither displayed nor collected for NavMeshes that are statically baked with the legacy system, nor for any patches recomputed locally due to the presence of NavMesh Obstacles.

See Also: NavMeshBuildSettings, NavMeshBuilder.BuildNavMeshData, NavMeshEditorHelpers.DrawBuildDebug.

Properties

flags	Specify which types of debug data to collect when building the NavMesh.

Description

Navigation mesh builder interface.

Note: There are two classes with the name `NavMeshBuilder`. They are declared in two different namespaces: `UnityEngine.AI` and `UnityEditor.AI`. Their methods are all listed together on this page. The namespace is indicated in parentheses next to each method.

Static Properties

isRunning	Returns true if an asynchronous build is still running. (UnityEditor)

Static Methods

BuildNavMesh	Build the Navmesh. (UnityEditor)
BuildNavMeshAsync	Build the Navmesh Asyncronously. (UnityEditor)
BuildNavMeshData	Builds a NavMesh data object from the provided input sources. (UnityEngine)
BuildNavMeshForMultipleScenes	Builds the combined navmesh for the contents of multiple Scenes. (UnityEditor)
Cancel	Cancel Navmesh construction. (UnityEditor) See Also: NavMeshBuilder.BuildNavMeshAsync
ClearAllNavMeshes	Clear all Navmeshes. (UnityEditor)
CollectSources	For convenience, you can create a list of build sources directly from the current geometry. (UnityEngine)
CollectSourcesInStage	Creates a list of build sources directly from the current geometry in the specified editor stage. (UnityEditor)
UpdateNavMeshData	Incrementally updates the NavMeshData based on the sources. (UnityEngine)
UpdateNavMeshDataAsync	Asynchronously and incrementally updates the NavMeshData based on the sources. (UnityEngine)

Description

The NavMesh build markup allows you to control how certain objects are treated during the NavMesh build process, specifically when collecting sources for building.

You can override the area type or specify that certain objects should be excluded from collected sources. The markup is applied hierarchically. See Also: NavMeshBuilder.CollectSources.

Properties

area	The area type to use when override area is enabled.
ignoreFromBuild	Use this to specify whether the GameObject and its children should be ignored.
overrideArea	Use this to specify whether the area type of the GameObject and its children should be overridden by the area type specified in this struct.
root	Use this to specify which GameObject (including the GameObject's children) the markup should be applied to.

Description

The NavMeshBuildSettings struct allows you to specify a collection of settings which describe the dimensions and limitations of a particular agent type.

You might want to define multiple NavMeshBuildSettings if your game involves characters with large differences in height, width or climbing ability.

You can also use this struct to control the precision and granularity of the build process, by setting the voxel and tile sizes. Some of the values are coupled, meaning there are constraints on the values based on other values. For example, it's not valid for agentClimb to be larger than agentHeight. To help diagnose violations of these rules, a special method ValidationReport can be evaluated.

Properties

agentClimb	The maximum vertical step size an agent can take.
agentHeight	The height of the agent for baking in world units.
agentRadius	The radius of the agent for baking in world units.
agentSlope	The maximum slope angle which is walkable (angle in degrees).
agentTypeID	The agent type ID the NavMesh will be baked for.
debug	Options for collecting debug data during the build process.
minRegionArea	The approximate minimum area of individual NavMesh regions.
overrideTileSize	Enables overriding the default tile size. See Also: tileSize.
overrideVoxelSize	Enables overriding the default voxel size. See Also: voxelSize.
tileSize	Sets the tile size in voxel units.
voxelSize	Sets the voxel size in world length units.

Public Methods

ValidationReport	Validates the properties of NavMeshBuildSettings.

Description

The input to the NavMesh builder is a list of NavMesh build sources.

Their shape can be one of the following: mesh, terrain, box, sphere, or capsule. Each of them is described by a NavMeshBuildSource struct.

You can specify a build source by filling a NavMeshBuildSource struct and adding that to the list of sources that are passed to the bake function. Alternatively, you can use the collect API to quickly create NavMesh build sources from available render meshes or physics colliders. See NavMeshBuilder.CollectSources.

If you use this function at runtime, any meshes with read/write access disabled will not be processed or included in the final NavMesh. See Mesh.isReadable.

```
using UnityEngine;
using UnityEngine.AI;


public class Example : MonoBehaviour
{
    // Make a build source for a box in local space
    public NavMeshBuildSource BoxSource10x10()
    {
        var src = new NavMeshBuildSource();
        src.transform = transform.localToWorldMatrix;
        src.shape = NavMeshBuildSourceShape.Box;
        src.size = new Vector3(10.0f, 0.1f, 10.0f);
        return src;
    }
}
```

Properties

area	Describes the area type of the NavMesh surface for this object.
component	Points to the owning component - if available, otherwise null.
shape	The type of the shape this source describes. See Also: NavMeshBuildSourceShape.
size	Describes the dimensions of the shape.
sourceObject	Describes the object referenced for Mesh and Terrain types of input sources.

transform	Describes the local to world transformation matrix of the build source. That is, position and orientation and scale of the shape.

Description

Contains and represents NavMesh data.

An object of this class can be used for creating instances of NavMeshes. See NavMesh.AddNavMeshData. The contained NavMesh can be built and updated using the build API. See NavMeshBuilder and methods therein.

Properties

position	Gets or sets the world space position of the NavMesh data.
rotation	Gets or sets the orientation of the NavMesh data.
sourceBounds	Returns the bounding volume of the input geometry used to build this NavMesh (Read Only).

Constructors

NavMeshData	Constructs a new object for representing a NavMesh for the default agent type.

Inherited Members

Properties

hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.

Public Methods

GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.

Static Methods

Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.

FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Description

The instance is returned when adding NavMesh data.

A valid NavMesh data instance is available to the navigation system. This means you can calculate paths etc. using that instance. You also need the instance if you want to remove the NavMesh data at a later time.

See Also: Remove, NavMesh.AddNavMeshData, NavMesh.RemoveNavMeshData.

Properties

owner	Get or set the owning Object.
valid	True if the NavMesh data is added to the navigation system - otherwise false (Read Only).

Public Methods

Remove	Removes this instance from the NavMesh system.

Description

Result information for NavMesh queries.

Properties

distance	Distance to the point of hit.
hit	Flag set when hit.
mask	Mask specifying NavMesh area at point of hit.
normal	Normal at the point of hit.
position	Position of hit.

Description

Used for runtime manipulation of links connecting polygons of the NavMesh.

A typical use case is to connect different navigation meshes. The link is described by the NavMeshLinkData struct. The NavMesh Link is the scripting equivalent of the class-OffMeshLink component – but allows for wide links as opposed to just points.

Properties

agentTypeID	Specifies which agent type this link is available for.
area	Area type of the link.
bidirectional	If true, the link can be traversed in both directions, otherwise only from start to end position.
costModifier	If positive, overrides the pathfinder cost to traverse the link.
endPosition	End position of the link.
startPosition	Start position of the link.
width	If positive, the link will be rectangle aligned along the line from start to end.

Description

An instance representing a link available for pathfinding.

It can also be used for removing NavMesh links from the navigation system.

Properties

owner	Get or set the owning Object.
valid	True if the NavMesh link is added to the navigation system - otherwise false (Read Only).

Public Methods

Remove	Removes this instance from the game.

Description

An obstacle for NavMeshAgents to avoid.

A NavMeshObstacle is cylindrical in shape and can move around the surface of the NavMesh with a specified velocity. By default, the obstacle will only affect the agent's *avoidance* behaviour rather than the pathfinding. This means that the agent will ignore the obstacle when plotting a path but will sidestep around it while moving along the path. If *carving* is enabled, the obstacle will create a temporary "hole" in the NavMesh. The hole will be recognised by the pathfinding, so paths will be plotted to avoid the obstacle. This means that if, say, an obstacle blocks a narrow gap, the pathfinding will seek an alternative route to the target. Without carving, the agent will head for the gap but won't be able to pass until the obstacle is clear.

See Also: NavMeshAgent.

Properties

carveOnlyStationary	Should this obstacle be carved when it is constantly moving?
carving	Should this obstacle make a cut-out in the navmesh.
carvingMoveThreshold	Threshold distance for updating a moving carved hole (when carving is enabled).
carvingTimeToStationary	Time to wait until obstacle is treated as stationary (when carving and carveOnlyStationary are enabled).
center	The center of the obstacle, measured in the object's local space.
height	Height of the obstacle's cylinder shape.
radius	Radius of the obstacle's capsule shape.
shape	The shape of the obstacle.
size	The size of the obstacle, measured in the object's local space.
velocity	Velocity at which the obstacle moves around the NavMesh.

Inherited Members

Properties

enabled	Enabled Behaviours are Updated, disabled Behaviours are not.
isActiveAndEnabled	Has the Behaviour had active and enabled called?
gameObject	The game object this component is attached to. A component is always attached to a game object.

tag	The tag of this game object.
transform	The Transform attached to this GameObject.
hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.

Public Methods

BroadcastMessage	Calls the method named methodName on every MonoBehaviour in this game object or any of its children.
CompareTag	Is this game object tagged with tag ?
GetComponent	Returns the component of Type type if the game object has one attached, null if it doesn't.
GetComponentInChildren	Returns the component of Type type in the GameObject or any of its children using depth first search.
GetComponentInParent	Returns the component of Type type in the GameObject or any of its parents.
GetComponents	Returns all components of Type type in the GameObject.
GetComponentsInChildren	Returns all components of Type type in the GameObject or any of its children.
GetComponentsInParent	Returns all components of Type type in the GameObject or any of its parents.
SendMessage	Calls the method named methodName on every MonoBehaviour in this game object.
SendMessageUpwards	Calls the method named methodName on every MonoBehaviour in this game object and on every ancestor of the behaviour.
GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.

Static Methods

Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.

Instantiate	Clones the object original and returns the clone.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Description

A path as calculated by the navigation system.

The path is represented as a list of waypoints stored in the corners array. These points are not set directly from user scripts but a NavMeshPath with points correctly assigned is returned by the NavMesh.CalculatePath function and the NavMeshAgent.path property.

Properties

corners	Corner points of the path. (Read Only)
status	Status of the path. (Read Only)

Constructors

NavMeshPath	NavMeshPath constructor.

Public Methods

ClearCorners	Erase all corner points from path.
GetCornersNonAlloc	Calculate the corners for the path.

Description

Specifies which agent type and areas to consider when searching the NavMesh.

This struct is used with the NavMesh query methods overloaded with the query filter argument. See Also: NavMesh.CalculatePath, NavMesh.Raycast, NavMesh.FindClosestEdge, NavMesh.SamplePosition.

Properties

agentTypeID	The agent type ID, specifying which navigation meshes to consider for the query functions.
areaMask	A bitmask representing the traversable area types.

Public Methods

GetAreaCost	Returns the area cost multiplier for the given area type for this filter.
SetAreaCost	Sets the pathfinding cost multiplier for this filter for a given area type.

Description

Contains data describing a triangulation of a navmesh.

Properties

areas	NavMesh area indices for the navmesh triangulation.
indices	Triangle indices for the navmesh triangulation.
vertices	Vertices for the navmesh triangulation.

Description

Link allowing movement outside the planar navigation mesh.

Properties

activated	Is link active.
area	NavMesh area index for this OffMeshLink component.
autoUpdatePositions	Automatically update endpoints.
biDirectional	Can link be traversed in both directions.
costOverride	Modify pathfinding cost for the link.
endTransform	The transform representing link end position.
occupied	Is link occupied. (Read Only)
startTransform	The transform representing link start position.

Public Methods

UpdatePositions	Explicitly update the link endpoints.

Inherited Members

Properties

enabled	Enabled Behaviours are Updated, disabled Behaviours are not.
isActiveAndEnabled	Has the Behaviour had active and enabled called?
gameObject	The game object this component is attached to. A component is always attached to a game object.
tag	The tag of this game object.
transform	The Transform attached to this GameObject.
hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.

Public Methods

BroadcastMessage	Calls the method named methodName on every MonoBehaviour in this game object or any of its children.
CompareTag	Is this game object tagged with tag ?
GetComponent	Returns the component of Type type if the game object has one attached, null if it doesn't.
GetComponentInChildren	Returns the component of Type type in the GameObject or any of its children using depth first search.
GetComponentInParent	Returns the component of Type type in the GameObject or any of its parents.
GetComponents	Returns all components of Type type in the GameObject.
GetComponentsInChildren	Returns all components of Type type in the GameObject or any of its children.
GetComponentsInParent	Returns all components of Type type in the GameObject or any of its parents.
SendMessage	Calls the method named methodName on every MonoBehaviour in this game object.
SendMessageUpwards	Calls the method named methodName on every MonoBehaviour in this game object and on every ancestor of the behaviour.
GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.

Static Methods

Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Description

State of OffMeshLink.

Properties

activated	Is link active (Read Only).
endPos	Link end world position (Read Only).
linkType	Link type specifier (Read Only).
offMeshLink	The OffMeshLink if the link type is a manually placed Offmeshlink (Read Only).
startPos	Link start world position (Read Only).
valid	Is link valid (Read Only).

Description

Bitmask used for operating with debug data from the NavMesh build process.

Used in two situations:
- within NavMeshBuildSettings.debug to specify which debug data to retain after the build process has completed, preserving the world position and orientation;
- as a parameter of NavMeshEditorHelpers.DrawBuildDebug to control which of the available debug data types to display for a specified NavMesh.

See Also: NavMeshBuildSettings.

Properties

None	No debug data from the NavMesh build process is taken into consideration.
InputGeometry	The triangles of all the geometry that is used as a base for computing the new NavMesh.
Voxels	The voxels produced by rasterizing the source geometry into walkable and unwalkable areas.
Regions	The segmentation of the traversable surfaces into smaller areas necessary for producing simple polygons.
RawContours	The contours that follow precisely the edges of each surface region.
SimplifiedContours	Contours bounding each of the surface regions, described through fewer vertices and straighter edges compared to RawContours.
PolygonMeshes	Meshes of convex polygons constructed within the unified contours of adjacent regions.
PolygonMeshesDetail	The triangulated meshes with height details that better approximate the source geometry.
All	All debug data from the NavMesh build process is taken into consideration.

Description

Used with NavMeshBuildSource to define the shape for building NavMesh.

Properties

Mesh	Describes a Mesh source for use with NavMeshBuildSource.
Terrain	Describes a TerrainData source for use with NavMeshBuildSource.
Box	Describes a box primitive for use with NavMeshBuildSource.
Sphere	Describes a sphere primitive for use with NavMeshBuildSource.
Capsule	Describes a capsule primitive for use with NavMeshBuildSource.
ModifierBox	Describes a ModifierBox source for use with NavMeshBuildSource.

Description

Used for specifying the type of geometry to collect. Used with NavMeshBuilder.CollectSources.

Properties

RenderMeshes	Collect meshes form the rendered geometry.
PhysicsColliders	Collect geometry from the 3D physics collision representation.

Description

Shape of the obstacle.

Properties

Capsule	Capsule shaped obstacle.
Box	Box shaped obstacle.

Description

Status of path.

Properties

PathComplete	The path terminates at the destination.
PathPartial	The path cannot reach the destination.
PathInvalid	The path is invalid.

Description

Level of obstacle avoidance.

Properties

NoObstacleAvoidance	Disable avoidance.
LowQualityObstacleAvoidance	Enable simple avoidance. Low performance impact.
MedQualityObstacleAvoidance	Medium avoidance. Medium performance impact.
GoodQualityObstacleAvoidance	Good avoidance. High performance impact.
HighQualityObstacleAvoidance	Enable highest precision. Highest performance impact.

Description

Link type specifier.

Properties

LinkTypeManual	Manually specified type of link.
LinkTypeDropDown	Vertical drop.
LinkTypeJumpAcross	Horizontal jump.

Description

Unity Analytics provides insight into your game users e.g. DAU, MAU.

Note: The documentation that follows is for usage of Unity Analytics with Unity 5.1 and above. To integrate Unity Analytics with earlier versions of Unity, refer to the documentation at https://analytics.cloud.unity3d.com/docs

Specifically, out-of-the-box features include access to user engagement and retention data through Metric Monitor and Data Explorer. Segmentation of players across various user dimensions and characteristics is automatically applied, with more than 20 standard segments available without any additional instrumentation.

For more detail around features, refer to the Unity Analytics homepage at http://unity3d.com/unity/analytics.

Static Properties

deviceStatsEnabled	Controls whether the sending of device stats at runtime is enabled.
enabled	Controls whether the Analytics service is enabled at runtime.
limitUserTracking	Controls whether to limit user tracking at runtime.

Static Methods

CustomEvent	Custom Events (optional).
FlushEvents	Attempts to flush immediately all queued analytics events to the network and filesystem cache if possible (optional).
RegisterEvent	This API is used for registering a Runtime Analytics event. It is meant for internal use only and is likely to change in the future. User code should never use this API.
SendEvent	This API is used to send a Runtime Analytics event. It is meant for internal use only and is likely to change in the future. User code should never use this API.
SetUserBirthYear	User Demographics (optional).
SetUserGender	User Demographics (optional).
SetUserId	User Demographics (optional).
Transaction	Tracking Monetization (optional).

Description

The main class of the Unity Analytics Standard Event SDK.

The event methods in this class provide 'fire and forget' ease of use, allowing you to send a standard event with a single line of code. If you expect to be sending the same standard event with some amount of frequency, you may want to create an event payload and invoke the `Send` method when the event occurs. The event methods in this class provide "fire and forget" ease of use, allowing you to send a standard event with a single line of code. If you expect to be sending the same standard event with some amount of frequency, you may want to create an event payload and invoke the `Send` method when the event occurs.

Static Properties

debugMode	Gets or sets a value indicating whether debug mode is enabled.
sdkVersion	Gets the Unity Analytics Standard Event SDK version.

Static Methods

AchievementStep	Sends a achievement_step event.
AchievementUnlocked	Sends a achievement_unlocked event.
AdComplete	Sends a ad_complete event.
AdOffer	Sends a ad_offer event.
AdSkip	Sends a ad_skip event.
AdStart	Sends a ad_start event.
ChatMessageSent	Sends a chat_message_sent event.
Custom	Sends a custom event (eventName) with data (eventData)
CustomEvent	Sends a custom event.
CutsceneSkip	Sends a cutscene_skip event.
CutsceneStart	Sends a cutscene_start event.
FirstInteraction	Sends a first_interaction event.
GameOver	Sends a game_over event.
GameStart	Sends a game_start event.
IAPTransaction	Sends a iap_transaction event.

ItemAcquired	Sends a item_acquired event.
ItemSpent	Sends a item_spent event.
LevelComplete	Sends a level_complete event.
LevelFail	Sends a level_fail event.
LevelQuit	Sends a level_quit event.
LevelSkip	Sends a level_skip event.
LevelStart	Sends a level_start event.
LevelUp	Sends a level_up event.
PostAdAction	Sends a post_ad_action event.
PushNotificationClick	Sends a push_notification_click event.
PushNotificationEnable	Sends a push_notification_enable event.
ScreenVisit	Sends a screen_visit event.
SocialShare	Sends a social_share event.
SocialShareAccept	Sends a social_share_accept event.
StoreItemClick	Sends a store_item_click event.
StoreOpened	Sends a store_opened event.
TutorialComplete	Sends a tutorial_complete event.
TutorialSkip	Sends a tutorial_skip event.
TutorialStart	Sends a tutorial_start event.
TutorialStep	Sends a tutorial_step event.
UserSignup	Sends a user_signup event.

Description

Provides access to the Analytics session information for the current game instance.

This class contains static methods for looking up Analytics session information. Note that cumulative statistics are reset if the player re-installs the game.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Analytics;
using System;


public class NewBehaviourScript : MonoBehaviour
{
    void Start()
    {
        Debug.Log("Start   " + AnalyticsSessionInfo.userId + " " + AnalyticsSess
        AnalyticsSessionInfo.sessionStateChanged += OnSessionStateChanged;
    }


    void OnSessionStateChanged(AnalyticsSessionState sessionState, long sessionI
    {
        Debug.Log("Call    " + AnalyticsSessionInfo.userId  + " " + sessionState
    }
}
```

Static Properties

sessionCount	The number of sessions played since the app was installed.
sessionElapsedTime	The time elapsed, in milliseconds, since the beginning of the current game session.
sessionFirstRun	Reports whether the current session is the first session since the player installed the game or application.
sessionId	A random, unique GUID identifying the current game or app session.
sessionState	The current state of the session.
userId	A random GUID uniquely identifying sessions played on the same instance of your

game or app.

Events

sessionStateChanged	Dispatched when the Analytics session state changes.

Delegates

SessionStateChanged	Dispatched when the Analytics session state changes.

Obsolete

Description

Automatically create and send analytics events based on a variety of triggers.

Set event name, parameter names, and automatically use parameter values either from properties of attached Components or from literal values. Send the instrumented event.

Inherited Members

Properties

enabled	Enabled Behaviours are Updated, disabled Behaviours are not.
isActiveAndEnabled	Has the Behaviour had active and enabled called?
gameObject	The game object this component is attached to. A component is always attached to a game object.
tag	The tag of this game object.
transform	The Transform attached to this GameObject.
runInEditMode	Allow a specific instance of a MonoBehaviour to run in edit mode (only available in the editor).
useGUILayout	Disabling this lets you skip the GUI layout phase.
hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.

Public Methods

BroadcastMessage	Calls the method named methodName on every MonoBehaviour in this game object or any of its children.
CompareTag	Is this game object tagged with tag ?
GetComponent	Returns the component of Type type if the game object has one attached, null if it doesn't.
GetComponentInChildren	Returns the component of Type type in the GameObject or any of its children using depth first search.
GetComponentInParent	Returns the component of Type type in the GameObject or any of its parents.
GetComponents	Returns all components of Type type in the GameObject.

GetComponentsInChildren	Returns all components of Type type in the GameObject or any of its children.
GetComponentsInParent	Returns all components of Type type in the GameObject or any of its parents.
SendMessage	Calls the method named methodName on every MonoBehaviour in this game object.
SendMessageUpwards	Calls the method named methodName on every MonoBehaviour in this game object and on every ancestor of the behaviour.
CancelInvoke	Cancels all Invoke calls on this MonoBehaviour.
Invoke	Invokes the method methodName in time seconds.
InvokeRepeating	Invokes the method methodName in time seconds, then repeatedly every repeatRate seconds.
IsInvoking	Is any invoke on methodName pending?
StartCoroutine	Starts a coroutine.
StopAllCoroutines	Stops all coroutines running on this behaviour.
StopCoroutine	Stops the first coroutine named methodName, or the coroutine stored in routine running on this behaviour.
GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.

Static Methods

print	Logs message to the Unity Console (identical to Debug.Log).
Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.

Operators

bool	Does the object exist?

operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Messages

Awake	Awake is called when the script instance is being loaded.
FixedUpdate	This function is called every fixed framerate frame, if the MonoBehaviour is enabled.
LateUpdate	LateUpdate is called every frame, if the Behaviour is enabled.
OnAnimatorIK	Callback for setting up animation IK (inverse kinematics).
OnAnimatorMove	Callback for processing animation movements for modifying root motion.
OnApplicationFocus	Sent to all GameObjects when the player gets or loses focus.
OnApplicationPause	Sent to all GameObjects when the application pauses.
OnApplicationQuit	Sent to all game objects before the application quits.
OnAudioFilterRead	If OnAudioFilterRead is implemented, Unity will insert a custom filter into the audio DSP chain.
OnBecameInvisible	OnBecameInvisible is called when the renderer is no longer visible by any camera.
OnBecameVisible	OnBecameVisible is called when the renderer became visible by any camera.
OnCollisionEnter	OnCollisionEnter is called when this collider/rigidbody has begun touching another rigidbody/collider.
OnCollisionEnter2D	Sent when an incoming collider makes contact with this object's collider (2D physics only).
OnCollisionExit	OnCollisionExit is called when this collider/rigidbody has stopped touching another rigidbody/collider.
OnCollisionExit2D	Sent when a collider on another object stops touching this object's collider (2D physics only).
OnCollisionStay	:ref::OnCollisionStay is called once per frame for every collider/rigidbody that is touching rigidbody/collider.
OnCollisionStay2D	Sent each frame where a collider on another object is touching this object's collider (2D physics only).
OnConnectedToServer	Called on the client when you have successfully connected to a server.

OnControllerColliderHit	OnControllerColliderHit is called when the controller hits a collider while performing a Move.
OnDestroy	Destroying the attached Behaviour will result in the game or Scene receiving OnDestroy.
OnDisable	This function is called when the behaviour becomes disabled.
OnDisconnectedFromServer	Called on the client when the connection was lost or you disconnected from the server.
OnDrawGizmos	Implement OnDrawGizmos if you want to draw gizmos that are also pickable and always drawn.
OnDrawGizmosSelected	Implement OnDrawGizmosSelected to draw a gizmo if the object is selected.
OnEnable	This function is called when the object becomes enabled and active.
OnFailedToConnect	Called on the client when a connection attempt fails for some reason.
OnFailedToConnectToMasterServer	Called on clients or servers when there is a problem connecting to the MasterServer.
OnGUI	OnGUI is called for rendering and handling GUI events.
OnJointBreak	Called when a joint attached to the same game object broke.
OnJointBreak2D	Called when a Joint2D attached to the same game object breaks.
OnMasterServerEvent	Called on clients or servers when reporting events from the MasterServer.
OnMouseDown	OnMouseDown is called when the user has pressed the mouse button while over the GUIElement or Collider.
OnMouseDrag	OnMouseDrag is called when the user has clicked on a GUIElement or Collider and is still holding down the mouse.
OnMouseEnter	Called when the mouse enters the GUIElement or Collider.
OnMouseExit	Called when the mouse is not any longer over the GUIElement or Collider.
OnMouseOver	Called every frame while the mouse is over the GUIElement or Collider.
OnMouseUp	OnMouseUp is called when the user has released the mouse button.
OnMouseUpAsButton	OnMouseUpAsButton is only called when the mouse is released over the same GUIElement or Collider as it was pressed.
OnNetworkInstantiate	Called on objects which have been network instantiated with

	Network.Instantiate.
OnParticleCollision	OnParticleCollision is called when a particle hits a Collider.
OnParticleSystemStopped	OnParticleSystemStopped is called when all particles in the system have died, and no new particles will be born. New particles cease to be created either after Stop is called, or when the duration property of a non-looping system has been exceeded.
OnParticleTrigger	OnParticleTrigger is called when any particles in a particle system meet the conditions in the trigger module.
OnPlayerConnected	Called on the server whenever a new player has successfully connected.
OnPlayerDisconnected	Called on the server whenever a player disconnected from the server.
OnPostRender	OnPostRender is called after a camera finished rendering the Scene.
OnPreCull	OnPreCull is called before a camera culls the Scene.
OnPreRender	OnPreRender is called before a camera starts rendering the Scene.
OnRenderImage	OnRenderImage is called after all rendering is complete to render image.
OnRenderObject	OnRenderObject is called after camera has rendered the Scene.
OnSerializeNetworkView	Used to customize synchronization of variables in a script watched by a network view.
OnServerInitialized	Called on the server whenever a Network.InitializeServer was invoked and has completed.
OnTransformChildrenChanged	This function is called when the list of children of the transform of the GameObject has changed.
OnTransformParentChanged	This function is called when the parent property of the transform of the GameObject has changed.
OnTriggerEnter	OnTriggerEnter is called when the Collider other enters the trigger.
OnTriggerEnter2D	Sent when another object enters a trigger collider attached to this object (2D physics only).
OnTriggerExit	OnTriggerExit is called when the Collider other has stopped touching the trigger.
OnTriggerExit2D	Sent when another object leaves a trigger collider attached to this object (2D physics only).
OnTriggerStay	OnTriggerStay is called once per physics update for every Collider other that is touching the trigger.

OnTriggerStay2D	Sent each frame where another object is within a trigger collider attached to this object (2D physics only).
OnValidate	This function is called when the script is loaded or a value is changed in the inspector (Called in the editor only).
OnWillRenderObject	OnWillRenderObject is called for each camera if the object is visible and not a UI element.
Reset	Reset to default values.
Start	Start is called on the frame when a script is enabled just before any of the Update methods are called the first time.
Update	Update is called every frame, if the MonoBehaviour is enabled.

Description

Unity Performace provides insight into your game performace.

Static Properties

enabled	Controls whether the Performance Reporting service is enabled at runtime.
graphicsInitializationFinishTime	Time taken to initialize graphics in microseconds, measured from application startup.

Description

The source through which an item, consumable, or currency was acquired.

Properties

None	No available source, or source unknown. Note: Use a string value if the acquisition source is not one of the elements defined in this enumeration.
Store	Purchased using currency or consumable resources.
Earned	Awarded through an achievement or other in-game interaction.
Promotion	Granted as a promotion of some in-game feature or through cross promotion.
Gift	Granted without motive other than good feelings.
RewardedAd	Granted as a reward for watching an advertisement.
TimedReward	Granted periodically.
SocialReward	Granted through social engagement.

Description

The type of currency (premium or soft) used to acquire the item.

See Also: AnalyticsEvent.ItemAcquired.

Properties

Soft	Not directly purchased with real-world money.
Premium	Purchased with real-world money.

Description

The list of advertising or mediation network service providers.

Use the members of this list when possible to promote consistency in your Analytics data. However, if the advertising network is not included in this list, pass the argument to the Standard Event function as a string.

See Also: AnalyticsEvent.AdComplete, AnalyticsEvent.AdOffer, AnalyticsEvent.AdSkip, AnalyticsEvent.AdStart, AnalyticsEvent.PostAdAction.

Properties

None	No available network, or network unknown. Note: Use a string value to indicate a network not defined in this enumeration.
Aarki	Aarki network.
AdAction	AdAction network.
AdapTv	AdapTv network.
Adcash	Adcash network.
AdColony	AdColony network.
AdMob	AdMob network.
AerServ	AerServ network.
Airpush	Airpush network.
Altrooz	Altrooz network.
Ampush	Ampush network.
AppleSearch	AppleSearch network.
AppLift	AppLift network.
AppLovin	AppLovin network.
Appnext	Appnext network.
AppNexus	AppNexus network.
Appoday	Appoday network.
Appodeal	Appodeal network.
AppsUnion	AppsUnion network.

Avazu	Avazu network.
BlueStacks	BlueStacks network.
Chartboost	Chartboost network.
ClickDealer	ClickDealer network.
CPAlead	CPAlead network.
CrossChannel	CrossChannel network.
CrossInstall	CrossInstall network.
Epom	Epom network.
Facebook	Facebook network.
Fetch	Fetch network.
Fiksu	Fiksu network.
Flurry	Flurry network.
Fuse	Fuse network.
Fyber	Fyber network.
Glispa	Glispa network.
Google	Google network.
GrowMobile	GrowMobile network.
HeyZap	HeyZap network.
HyperMX	HyperMX network.
Iddiction	Iddiction network.
IndexExchange	IndexExchange network.
InMobi	InMobi network.
Instagram	Instagram network.
Instal	Instal network.
Ipsos	Ipsos network.
IronSource	IronSource network.
Jirbo	Jirbo network.
Kimia	Kimia network.

Leadbolt	Leadbolt network.
Liftoff	Liftoff network.
Manage	Manage network.
Matomy	Matomy network.
MediaBrix	MediaBrix network.
MillenialMedia	MillenialMedia network.
Minimob	Minimob network.
MobAir	MobAir network.
MobileCore	MobileCore network.
Mobobeat	Mobobeat network.
Mobusi	Mobusi network.
Mobvista	Mobvista network.
MoPub	MoPub network.
Motive	Motive network.
Msales	Msales network.
NativeX	NativeX network.
OpenX	OpenX network.
Pandora	Pandora network.
PropellerAds	PropellerAds network.
Revmob	Revmob network.
RubiconProject	RubiconProject network.
SiriusAd	SiriusAd network.
Smaato	Smaato network.
SponsorPay	SponsorPay network.
SpotXchange	SpotXchange network.
StartApp	StartApp network.
Tapjoy	Tapjoy network.
Taptica	Taptica network.

Tremor	Tremor network.
TrialPay	TrialPay network.
Twitter	Twitter network.
UnityAds	UnityAds network.
Vungle	Vungle network.
Yeahmobi	Yeahmobi network.
YuMe	YuMe network.

Description

Analytics API result.

Properties

Ok	Analytics API result: Success.
NotInitialized	Analytics API result: Analytics not initialized.
AnalyticsDisabled	Analytics API result: Analytics is disabled.
TooManyItems	Analytics API result: Too many parameters.
SizeLimitReached	Analytics API result: Argument size limit.
TooManyRequests	Analytics API result: Too many requests.
InvalidData	Analytics API result: Invalid argument value.
UnsupportedPlatform	Analytics API result: This platform doesn't support Analytics.

Description

Session tracking states.

Properties

kSessionStopped	Session tracking has stopped.
kSessionStarted	Session tracking has started.
kSessionPaused	Session tracking has paused.
kSessionResumed	Session tracking has resumed.

Description

The list of authorization networks for in-game login sessions.

Use the members of this list when possible to promote consistency in your Analytics data. However, if the authorization network is not included in this list, pass the argument to the Standard Event function as a string.

See Also: AnalyticsEvent.UserSignup.

Properties

None	No available authorization network, or network unknown. Note: Use a string value to indicate an authorization network not defined in this enumeration.
Internal	You use your own authorization system.
Facebook	Facebook authorization.
Twitter	Twitter authorization.
Google	Google authorization.
GameCenter	GameCenter authorization.

Description

User Demographics: Gender of a user.

Properties

Male	User Demographics: Male Gender of a user.
Female	User Demographics: Female Gender of a user.
Unknown	User Demographics: Unknown Gender of a user.

Description

The list of the in-game screen or interface types.

Use the members of this list when possible to promote consistency in your Analytics data. However, if the screen name is not included in this list, pass the argument to the Standard Event function as a string.

See Also: AnalyticsEvent.ScreenVisit.

Properties

None	No defined screen name. Note: Use a string value to indicate a screen name not defined in this enumeration.
MainMenu	The game's main menu screen.
Settings	The game's settings screen.
Map	The game's map screen.
Lose	The screen shown when the player loses the game.
Win	The screen shown when the player wins the game.
Credits	The game's credits screen.
Title	The game's title screen.
IAPPromo	The screen shown for IAP promotions.
CrossPromo	The screen shown for cross promotions.
FeaturePromo	The screen shown for feature promotions.
Hint	The game's hint screen.
Pause	The game's pause screen.
Inventory	The game's inventory screen.
Leaderboard	The game's leaderboard screen.
Achievements	The game's achievement screen.
Lobby	The game's lobby screen.

Description

The list of media types or modes used for social engagements.

Use the members of this list when possible to promote consistency in your Analytics data. However, if the media type or mode is not included in this list, pass the argument to the Standard Event function as a string.

See Also: AnalyticsEvent.SocialShare, AnalyticsEvent.SocialShareAccept.

Properties

None	No defined engagement media or mode. Note: Use a string value to indicate a mode of social engagement not defined in this enumeration.
TextOnly	A text-based mode of engagement, such as a text or chat message.
Image	An image-based mode of engagement.
Video	A video-based mode of engagement.
Invite	A mode of engagement that involves an invitation.
Achievement	An in-game achievement.

Description

The list of social networks used with social engagements.

Use the members of this list when possible to promote consistency in your Analytics data. However, if the social network is not included in this list, pass the argument to the Standard Event function as a string.

See Also: AnalyticsEvent.SocialShare, AnalyticsEvent.SocialShareAccept.

Properties

None	No available social network, or network unknown. Note: Use a string value to indicate social network not defined in this enumeration.
Facebook	Facebook.
Twitter	Twitter.
Instagram	Instagram.
GooglePlus	Google Plus.
Pinterest	Pinterest.
WeChat	WeChat.
SinaWeibo	The Sina Weibo social network.
TencentWeibo	The Tencent Weibo social network.
QQ	QQ and QZone networks.
Zhihu	Zhihu network.
VK	VKontakte network.
OK_ru	Odnoklassniki network.

Description

The list of in-game store categories.

See Also: AnalyticsEvent.StoreItemClick, AnalyticsEvent.StoreOpened

Properties

Soft	Contains items purchasable with in-game currency only.
Premium	Contains items purchasable with real-world money.

Implements interfaces:IConstraint

Description

Constrains the orientation of an object relative to the position of one or more source objects, such that the object is facing the average position of the sources.

Properties

aimVector	The axis towards which the constrained object orients.
constraintActive	Activates or deactivates the constraint.
locked	Locks the offset and rotation at rest.
rotationAtRest	The rotation used when the sources have a total weight of 0.
rotationAxis	The axes affected by the AimConstraint.
rotationOffset	Represents an offset from the constrained orientation.
sourceCount	The number of sources set on the component (read-only).
upVector	The up vector.
weight	The weight of the constraint component.
worldUpObject	The world up object, used to calculate the world up vector when the world up Type is AimConstraint.WorldUpType.ObjectUp or AimConstraint.WorldUpType.ObjectRotationUp.
worldUpType	The type of the world up vector.
worldUpVector	The world up Vector used when the world up type is AimConstraint.WorldUpType.Vector or AimConstraint.WorldUpType.ObjectRotationUp.

Public Methods

AddSource	Adds a constraint source.
GetSource	Gets a constraint source by index.
GetSources	Gets the list of sources.
RemoveSource	Removes a source from the component.
SetSource	Sets a source at a specified index.
SetSources	Sets the list of sources on the component.

Inherited Members

Properties

enabled	Enabled Behaviours are Updated, disabled Behaviours are not.
isActiveAndEnabled	Has the Behaviour had active and enabled called?
gameObject	The game object this component is attached to. A component is always attached to a game object.
tag	The tag of this game object.
transform	The Transform attached to this GameObject.
hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.

Public Methods

BroadcastMessage	Calls the method named methodName on every MonoBehaviour in this game object or any of its children.
CompareTag	Is this game object tagged with tag ?
GetComponent	Returns the component of Type type if the game object has one attached, null if it doesn't.
GetComponentInChildren	Returns the component of Type type in the GameObject or any of its children using depth first search.
GetComponentInParent	Returns the component of Type type in the GameObject or any of its parents.
GetComponents	Returns all components of Type type in the GameObject.
GetComponentsInChildren	Returns all components of Type type in the GameObject or any of its children.
GetComponentsInParent	Returns all components of Type type in the GameObject or any of its parents.
SendMessage	Calls the method named methodName on every MonoBehaviour in this game object.
SendMessageUpwards	Calls the method named methodName on every MonoBehaviour in this game object and on every ancestor of the behaviour.
GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.

Static Methods

Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Description

Specifies how the world up vector used by the aim constraint is defined.

Properties

SceneUp	Uses and defines the world up vector as the Unity Scene up vector (the Y axis).
ObjectUp	Uses and defines the world up vector as a vector from the constrained object, in the direction of the up object.
ObjectRotationUp	Uses and defines the world up vector as relative to the local space of the object.
Vector	Uses and defines the world up vector as a vector specified by the user.
None	Neither defines nor uses a world up vector.

Implements interfaces:IPlayable

Description

A Playable that controls an AnimationClip.

NOTE: You can use PlayableExtensions methods with AnimationClipPlayable objects.

Public Methods

GetAnimationClip	Returns the AnimationClip stored in the AnimationClipPlayable.
GetApplyFootIK	Returns the state of the ApplyFootIK flag.
GetApplyPlayableIK	Returns the state of the ApplyPlayableIK flag.
SetApplyFootIK	Sets the value of the ApplyFootIK flag.
SetApplyPlayableIK	Requests OnAnimatorIK to be called on the animated GameObject.

Static Methods

Create	Creates an AnimationClipPlayable in the PlayableGraph.

Implements interfaces:IPlayable

Description

An implementation of IPlayable that controls an animation layer mixer.

NOTE: You can use PlayableExtensions methods with AnimationLayerMixerPlayable objects.

Static Properties

Null	Returns an invalid AnimationLayerMixerPlayable.

Public Methods

IsLayerAdditive	Returns true if the layer is additive, false otherwise.
SetLayerAdditive	Specifies whether a layer is additive or not. Additive layers blend with previous layers.
SetLayerMaskFromAvatarMask	Sets the mask for the current layer.

Static Methods

Create	Creates an AnimationLayerMixerPlayable in the PlayableGraph.

Implements interfaces:IPlayable

Description

An implementation of IPlayable that controls an animation mixer.

NOTE: You can use PlayableExtensions methods with AnimationMixerPlayable objects.

Static Properties

Null	Returns an invalid AnimationMixerPlayable.

Static Methods

Create	Creates an AnimationMixerPlayable in the PlayableGraph.

Description

A PlayableBinding that contains information representing an AnimationPlayableOutput.

Static Methods

Create	Creates a PlayableBinding that contains information representing an AnimationPlayableOutput.

Implements interfaces:IPlayableOutput

Description

A IPlayableOutput implementation that connects the PlayableGraph to an Animator in the Scene.

NOTE: You can use PlayableOutputExtensions methods on AnimationPlayableOutput objects.

Public Methods

GetTarget	Returns the Animator that plays the animation graph.
SetTarget	Sets the Animator that plays the animation graph.

Static Methods

Create	Creates an AnimationPlayableOutput in the PlayableGraph.

Implements interfaces:IPlayable

Description

An implementation of IPlayable that controls an animation RuntimeAnimatorController.

NOTE: You can use PlayableExtensions methods with AnimatorControllerPlayable objects.

Static Properties

Null	Returns an invalid AnimatorControllerPlayable.

Static Methods

Create	Creates an AnimatorControllerPlayable in the PlayableGraph.

Description

Represents a source for the constraint.

Properties

sourceTransform	The transform component of the source object.
weight	The weight of the source in the evaluation of the constraint.

Implements interfaces:IConstraint

Description

Constrains the orientation of an object relative to the position of one or more source objects, such that the object is facing the average position of the sources. The LookAtConstraint is a simplified AimConstraint typically used with a Camera.

Properties

constraintActive	Activates or deactivates the constraint.
locked	Locks the offset and rotation at rest.
roll	The rotation angle along the z axis of the object. The constraint uses this property to calculate the world up vector when Animations.LookAtConstraint.UseUpObject is false.
rotationAtRest	The rotation used when the sources have a total weight of 0.
rotationOffset	Represents an offset from the constrained orientation.
sourceCount	The number of sources set on the component (Read Only).
useUpObject	Determines how the up vector is calculated.
weight	The weight of the constraint component.
worldUpObject	The world up object, used to calculate the world up vector when Animations.LookAtConstraint.UseUpObject is true.

Public Methods

AddSource	Adds a constraint source.
GetSource	Gets a constraint source by index.
GetSources	Gets the list of sources.
RemoveSource	Removes a source from the component.
SetSource	Sets a source at a specified index.
SetSources	Sets the list of sources on the component.

Inherited Members

Properties

enabled	Enabled Behaviours are Updated, disabled Behaviours are not.
isActiveAndEnabled	Has the Behaviour had active and enabled called?
gameObject	The game object this component is attached to. A component is always attached to a game object.
tag	The tag of this game object.
transform	The Transform attached to this GameObject.
hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.

Public Methods

BroadcastMessage	Calls the method named methodName on every MonoBehaviour in this game object or any of its children.
CompareTag	Is this game object tagged with tag ?
GetComponent	Returns the component of Type type if the game object has one attached, null if it doesn't.
GetComponentInChildren	Returns the component of Type type in the GameObject or any of its children using depth first search.
GetComponentInParent	Returns the component of Type type in the GameObject or any of its parents.
GetComponents	Returns all components of Type type in the GameObject.
GetComponentsInChildren	Returns all components of Type type in the GameObject or any of its children.
GetComponentsInParent	Returns all components of Type type in the GameObject or any of its parents.
SendMessage	Calls the method named methodName on every MonoBehaviour in this game object.
SendMessageUpwards	Calls the method named methodName on every MonoBehaviour in this game object and on every ancestor of the behaviour.
GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.

Static Methods

Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy

	instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Implements interfaces:IConstraint

Description

Constrains the orientation and translation of an object to one or more source objects. The constrained object behaves as if it is in the hierarchy of the sources.

Properties

constraintActive	Activates or deactivates the constraint.
locked	Locks the offsets and position (translation and rotation) at rest.
rotationAtRest	The rotation used when the sources have a total weight of 0.
rotationAxis	The rotation axes affected by the ParentConstraint.
rotationOffsets	The rotation offsets from the constrained orientation.
sourceCount	The number of sources set on the component (read-only).
translationAtRest	The position of the object in local space, used when the sources have a total weight of 0.
translationAxis	The translation axes affected by the ParentConstraint.
translationOffsets	The translation offsets from the constrained orientation.
weight	The weight of the constraint component.

Public Methods

AddSource	Adds a constraint source.
GetRotationOffset	Gets the rotation offset associated with a source by index.
GetSource	Gets a constraint source by index.
GetSources	Gets the list of sources.
GetTranslationOffset	Gets the rotation offset associated with a source by index.
RemoveSource	Removes a source from the component.
SetRotationOffset	Sets the rotation offset associated with a source by index.
SetSource	Sets a source at a specified index.
SetSources	Sets the list of sources on the component.
SetTranslationOffset	Sets the translation offset associated with a source by index.

Inherited Members

Properties

enabled	Enabled Behaviours are Updated, disabled Behaviours are not.
isActiveAndEnabled	Has the Behaviour had active and enabled called?
gameObject	The game object this component is attached to. A component is always attached to a game object.
tag	The tag of this game object.
transform	The Transform attached to this GameObject.
hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.

Public Methods

BroadcastMessage	Calls the method named methodName on every MonoBehaviour in this game object or any of its children.
CompareTag	Is this game object tagged with tag ?
GetComponent	Returns the component of Type type if the game object has one attached, null if it doesn't.
GetComponentInChildren	Returns the component of Type type in the GameObject or any of its children using depth first search.
GetComponentInParent	Returns the component of Type type in the GameObject or any of its parents.
GetComponents	Returns all components of Type type in the GameObject.
GetComponentsInChildren	Returns all components of Type type in the GameObject or any of its children.
GetComponentsInParent	Returns all components of Type type in the GameObject or any of its parents.
SendMessage	Calls the method named methodName on every MonoBehaviour in this game object.
SendMessageUpwards	Calls the method named methodName on every MonoBehaviour in this game object and on every ancestor of the behaviour.
GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.

Static Methods

Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Implements interfaces:IConstraint

Description

Constrains the position of an object relative to the position of one or more source objects.

Properties

constraintActive	Activates or deactivates the constraint.
locked	Locks the offset and position at rest.
sourceCount	The number of sources set on the component (read-only).
translationAtRest	The translation used when the sources have a total weight of 0.
translationAxis	The axes affected by the PositionConstraint.
translationOffset	The offset from the constrained position.
weight	The weight of the constraint component.

Public Methods

AddSource	Adds a constraint source.
GetSource	Gets a constraint source by index.
GetSources	Gets the list of sources.
RemoveSource	Removes a source from the component.
SetSource	Sets a source at a specified index.
SetSources	Sets the list of sources on the component.

Inherited Members

Properties

enabled	Enabled Behaviours are Updated, disabled Behaviours are not.
isActiveAndEnabled	Has the Behaviour had active and enabled called?
gameObject	The game object this component is attached to. A component is always attached to a game object.
tag	The tag of this game object.

transform	The Transform attached to this GameObject.
hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.

Public Methods

BroadcastMessage	Calls the method named methodName on every MonoBehaviour in this game object or any of its children.
CompareTag	Is this game object tagged with tag ?
GetComponent	Returns the component of Type type if the game object has one attached, null if it doesn't.
GetComponentInChildren	Returns the component of Type type in the GameObject or any of its children using depth first search.
GetComponentInParent	Returns the component of Type type in the GameObject or any of its parents.
GetComponents	Returns all components of Type type in the GameObject.
GetComponentsInChildren	Returns all components of Type type in the GameObject or any of its children.
GetComponentsInParent	Returns all components of Type type in the GameObject or any of its parents.
SendMessage	Calls the method named methodName on every MonoBehaviour in this game object.
SendMessageUpwards	Calls the method named methodName on every MonoBehaviour in this game object and on every ancestor of the behaviour.
GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.

Static Methods

Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.

Instantiate	Clones the object original and returns the clone.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Implements interfaces:IConstraint

Description

Constrains the rotation of an object relative to the rotation of one or more source objects.

Properties

constraintActive	Activates or deactivates the constraint.
locked	Locks the offset and rotation at rest.
rotationAtRest	The rotation used when the sources have a total weight of 0.
rotationAxis	The axes affected by the RotationConstraint.
rotationOffset	The offset from the constrained rotation.
sourceCount	The number of sources set on the component (read-only).
weight	The weight of the constraint component.

Public Methods

AddSource	Adds a constraint source.
GetSource	Gets a constraint source by index.
GetSources	Gets the list of sources.
RemoveSource	Removes a source from the component.
SetSource	Sets a source at a specified index.
SetSources	Sets the list of sources on the component.

Inherited Members

Properties

enabled	Enabled Behaviours are Updated, disabled Behaviours are not.
isActiveAndEnabled	Has the Behaviour had active and enabled called?
gameObject	The game object this component is attached to. A component is always attached to a game object.
tag	The tag of this game object.

transform	The Transform attached to this GameObject.
hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.

Public Methods

BroadcastMessage	Calls the method named methodName on every MonoBehaviour in this game object or any of its children.
CompareTag	Is this game object tagged with tag ?
GetComponent	Returns the component of Type type if the game object has one attached, null if it doesn't.
GetComponentInChildren	Returns the component of Type type in the GameObject or any of its children using depth first search.
GetComponentInParent	Returns the component of Type type in the GameObject or any of its parents.
GetComponents	Returns all components of Type type in the GameObject.
GetComponentsInChildren	Returns all components of Type type in the GameObject or any of its children.
GetComponentsInParent	Returns all components of Type type in the GameObject or any of its parents.
SendMessage	Calls the method named methodName on every MonoBehaviour in this game object.
SendMessageUpwards	Calls the method named methodName on every MonoBehaviour in this game object and on every ancestor of the behaviour.
GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.

Static Methods

Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.

Instantiate	Clones the object original and returns the clone.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Implements interfaces:IConstraint

Description

Constrains the scale of an object relative to the scale of one or more source objects.

Properties

constraintActive	Activates or deactivates the constraint.
locked	Locks the offset and scale at rest.
scaleAtRest	The scale used when the sources have a total weight of 0.
scaleOffset	The offset from the constrained scale.
scalingAxis	The axes affected by the ScaleConstraint.
sourceCount	The number of sources set on the component (read-only).
weight	The weight of the constraint component.

Public Methods

AddSource	Adds a constraint source.
GetSource	Gets a constraint source by index.
GetSources	Gets the list of sources.
RemoveSource	Removes a source from the component.
SetSource	Sets a source at a specified index.
SetSources	Sets the list of sources on the component.

Inherited Members

Properties

enabled	Enabled Behaviours are Updated, disabled Behaviours are not.
isActiveAndEnabled	Has the Behaviour had active and enabled called?
gameObject	The game object this component is attached to. A component is always attached to a game object.
tag	The tag of this game object.

transform	The Transform attached to this GameObject.
hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.

Public Methods

BroadcastMessage	Calls the method named methodName on every MonoBehaviour in this game object or any of its children.
CompareTag	Is this game object tagged with tag ?
GetComponent	Returns the component of Type type if the game object has one attached, null if it doesn't.
GetComponentInChildren	Returns the component of Type type in the GameObject or any of its children using depth first search.
GetComponentInParent	Returns the component of Type type in the GameObject or any of its parents.
GetComponents	Returns all components of Type type in the GameObject.
GetComponentsInChildren	Returns all components of Type type in the GameObject or any of its children.
GetComponentsInParent	Returns all components of Type type in the GameObject or any of its parents.
SendMessage	Calls the method named methodName on every MonoBehaviour in this game object.
SendMessageUpwards	Calls the method named methodName on every MonoBehaviour in this game object and on every ancestor of the behaviour.
GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.

Static Methods

Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.

Instantiate	Clones the object original and returns the clone.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Description

The common interface for constraint components.

Properties

constraintActive	Activate or deactivate the constraint.
locked	Lock or unlock the offset and position at rest.
sourceCount	Gets the number of sources currently set on the component.
weight	The weight of the constraint component.

Public Methods

AddSource	Add a constraint source.
GetSource	Gets a constraint source by index.
GetSources	Gets the list of sources.
RemoveSource	Removes a source from the component.
SetSource	Sets a source at a specified index.
SetSources	Sets the list of sources on the component.

Description

Represents the axes used in 3D space.

Properties

None	Represents the case when no axis is specified.
X	Represents the X axis.
Y	Represents the Y axis.
Z	Represents the Z axis.

Description

ReplayKit is only available on certain iPhone, iPad and iPod Touch devices running iOS 9.0 or later.

ReplayKit allows you to record the audio and video of your game, along with user commentary through the microphone. Start a recording with the StartRecording() function, and stop it with StopRecording(). You can preview the recording with the Preview() function, which launches a separate video viewer.

Note: The exact Apple devices which have support for ReplayKit are:
1. iPhone 5S or later
2. iPad mini 2 or later
3. iPod Touch 6th generation
4. iPad Air or later.

```
using System;
using UnityEngine;
#if PLATFORM_IOS
using UnityEngine.iOS;
using UnityEngine.Apple.ReplayKit;


public class Replay : MonoBehaviour
{
    string lastError = "";
    void OnGUI()
    {
        if (!ReplayKit.APIAvailable)
        {
            return;
        }
        var recording = ReplayKit.isRecording;
        string caption = recording ? "Stop Recording" : "Start Recording";
        if (GUI.Button(new Rect(10, 10, 500, 200), caption))
        {
            try
            {
                recording = !recording;
                if (recording)
                {
                    ReplayKit.StartRecording();
                }
                else
                {
```

```
                    ReplayKit.StopRecording();
                }
            }
            catch (Exception e)
            {
                lastError = e.ToString();
            }
        }


        GUI.Label(new Rect(10, 220, 500, 50), "Last error: " + ReplayKit.lastErr
        GUI.Label(new Rect(10, 280, 500, 50), "Last exception: " + lastError);


        if (ReplayKit.recordingAvailable)
        {
            if (GUI.Button(new Rect(10, 350, 500, 200), "Preview"))
            {
                ReplayKit.Preview();
            }
            if (GUI.Button(new Rect(10, 560, 500, 200), "Discard"))
            {
                ReplayKit.Discard();
            }
        }
    }
}
#endif
```

Static Properties

APIAvailable	A boolean that indicates whether the ReplayKit API is available (where True means available). (Read Only)
broadcastingAPIAvailable	A Boolean that indicates whether ReplayKit broadcasting API is available (true means available) (Read Only). Check the value of this property before making ReplayKit broadcasting API calls. On iOS versions prior to iOS 10, this property will have a value of false.
broadcastURL	A string property that contains an URL used to redirect the user to an on-going or completed broadcast (Read Only).
cameraEnabled	Camera enabled status, true, if camera enabled, false otherwise.
isBroadcasting	Boolean property that indicates whether a broadcast is currently in progress (Read Only).

isRecording	A boolean that indicates whether ReplayKit is making a recording (where True means a recording is in progress). (Read Only)
lastError	A string value of the last error incurred by the ReplayKit: Either 'Failed to get Screen Recorder' or 'No recording available'. (Read Only)
microphoneEnabled	Microphone enabled status, true, if microhone enabled, false otherwise.
recordingAvailable	A boolean value that indicates that a new recording is available for preview (where True means available). (Read Only)

Static Methods

Discard	Discard the current recording.
HideCameraPreview	Hide the camera preview view.
Preview	Preview the current recording
ShowCameraPreviewAt	Shows camera preview at coordinates posX and posY.
StartBroadcasting	Initiates and starts a new broadcast When StartBroadcast is called, user is presented with a broadcast provider selection screen, and then a broadcast setup screen. Once it is finished, a broadcast will be started, and the callback will be invoked. It will also be invoked in case of any error.
StartRecording	Start a new recording.
StopBroadcasting	Stops current broadcast. Will terminate currently on-going broadcast. If no broadcast is in progress, does nothing.
StopRecording	Stop the current recording.

Delegates

BroadcastStatusCallback	Function called at the completion of broadcast startup.

Description

A float comparer used by Assert performing approximate comparison.

Static Properties

kEpsilon	Default epsilon used by the comparer.
s_ComparerWithDefaultTolerance	Default instance of a comparer class with deafult error epsilon and absolute error check.

Constructors

FloatComparer	Creates an instance of the comparer.

Static Methods

AreEqual	Performs equality check with absolute error check.
AreEqualRelative	Performs equality check with relative error check.

Obsolete

Description

An extension class that serves as a wrapper for the Assert class.

The MustExtension class wraps the Assertion.Assert into a more readable and shorter syntax.

```
using UnityEngine;
using UnityEngine.Assertions;
using UnityEngine.Assertions.Must;


public class ExampleClass : MonoBehaviour
{
    public int health;
    public GameObject go;
    void Update()
    {
        //This assertion
        Assert.AreNotEqual(0, health);
        //can be also written like this
        health.MustNotBeEqual(0);


        //Same with this assertion
        Assert.IsTrue(go.activeInHierarchy);
        //can be also written like this
        go.activeInHierarchy.MustBeTrue();
    }
}
```

Description

The Assert class contains assertion methods for setting invariants in the code.

All method calls will be conditionally included only in the development builds, unless explicitly specified (see BuildOptions.ForceEnableAssertions). The inclusion of the assertions is controlled by UNITY_ASSERTIONS define.

A failure of an assertion method does not break the control flow of the execution. On a failure, an assertion message is logged (LogType.Assert) and the execution continues. If Assert.raiseExceptions is set to true, an AssertionException is thrown instead of logging a message.

If a debugger is attached to the project (System.Diagnostics.Debugger.IsAttached is true), AssertionException will be thrown in order to pause the excecution and invoke the debugger.

```
using UnityEngine;
using UnityEngine.Assertions;


public class AssertionExampleClass : MonoBehaviour
{
    public int health;
    public GameObject go;


    void Update()
    {
        // You expect the health never to be equal to zero
        Assert.AreNotEqual(0, health);


        // The referenced GameObject should be always (in every frame) be active
        Assert.IsTrue(go.activeInHierarchy);
    }
}
```

Static Properties

raiseExceptions	Whether Unity should throw an exception on a failure.

Static Methods

AreApproximatelyEqual	Asserts that the values are approximately equal. An absolute error check is used for approximate equality check (|a-b| < tolerance). Default tolerance is 0.00001f.Note: Every time you call the method with tolerance specified, a new instance of FloatComparer is created. For performance reasons you might want to instance your own comparer and pass it to the AreEqual method. If the tolerance is not specifies, a default comparer is used and the issue does not occur.
AreEqual	Asserts that the values are equal. If no comparer is specified, EqualityComparer<T>.Default is used.
AreNotApproximatelyEqual	Asserts that the values are approximately not equal. An absolute error check is used for approximate equality check (|a-b| < tolerance). Default tolerance is 0.00001f.
AreNotEqual	Asserts that the values are not equal.
IsFalse	Asserts that the condition is false.
IsNotNull	Asserts that the value is not null.
IsNull	Asserts that the value is null.
IsTrue	Asserts that the condition is true.

Description

An exception that is thrown on a failure. Assert.raiseExceptions needs to be set to true.

Implements interfaces:IPlayable

Description

An implementation of IPlayable that controls an AudioClip.

NOTE: You can use PlayableExtensions methods with AudioClipPlayable objects.

Static Methods

Create	Creates an AudioClipPlayable in the PlayableGraph.

Description

AudioMixer asset.

This is a singleton representing a specific audio mixer asset in the project.

Properties

outputAudioMixerGroup	Routing target.
updateMode	How time should progress for this AudioMixer. Used during Snapshot transitions.

Public Methods

ClearFloat	Resets an exposed parameter to its initial value.
FindMatchingGroups	Connected groups in the mixer form a path from the mixer's master group to the leaves. This path has the format "Master Group/Child of Master Group/Grandchild of Master Group", so to find the grandchild group in this example, a valid search string would be for instance "randchi" which would return exactly one group while "hild" or "oup/" would return 2 different groups.
FindSnapshot	The name must be an exact match.
GetFloat	Returns the value of the exposed parameter specified. If the parameter doesn't exist the function returns false. Prior to calling SetFloat and after ClearFloat has been called on this parameter the value returned will be that of the current snapshot or snapshot transition.
SetFloat	Sets the value of the exposed parameter specified. When a parameter is exposed, it is not controlled by mixer snapshots and can therefore only be changed via this function.
TransitionToSnapshots	Transitions to a weighted mixture of the snapshots specified. This can be used for games that specify the game state as a continuum between states or for interpolating snapshots from a triangulated map location.

Inherited Members

Properties

hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.

Public Methods

GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.

Static Methods

Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Description

Object representing a group in the mixer.

Inherited Members

Properties

hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.

Public Methods

GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.

Static Methods

Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Implements interfaces:IPlayable

Description

An implementation of IPlayable that controls an audio mixer.

NOTE: You can use PlayableExtensions methods with AudioMixerPlayable objects.

Description

Object representing a snapshot in the mixer.

Public Methods

TransitionTo	Performs an interpolated transition towards this snapshot over the time interval specified.

Inherited Members

Properties

hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.

Public Methods

GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.

Static Methods

Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Description

A PlayableBinding that contains information representing an AudioPlayableOutput.

Static Methods

Create	Creates a PlayableBinding that contains information representing an AudioPlayableOutput.

Implements interfaces:IPlayableOutput

Description

A IPlayableOutput implementation that will be used to play audio.

NOTE: You can use PlayableOutputExtensions methods with AudioPlayableOutput objects.

Static Properties

Null	Returns an invalid AudioPlayableOutput.

Public Methods

GetEvaluateOnSeek	Gets the state of output playback when seeking.
SetEvaluateOnSeek	Controls whether the output should play when seeking.

Static Methods

Create	Creates an AudioPlayableOutput in the PlayableGraph.

Description

The mode in which an AudioMixer should update its time.

Use this with AudioMixer.updateMode to define how the AudioMixer time is updated.

Properties

Normal	Update the AudioMixer with scaled game time.
UnscaledTime	Update the AudioMixer with unscaled realtime.

Description

Engine API for CrashReporting Service.

Static Properties

enableCaptureExceptions	This Boolean field will cause CrashReportHandler to capture exceptions when set to true. By default enable capture exceptions is true.
logBufferSize	The Performance Reporting service will keep a buffer of up to the last X log messages (Debug.Log, etc) to send along with crash reports. The default is 10 log messages, the max is 50. Set this to 0 to disable capture of logs with your crash reports.

Static Methods

GetUserMetadata	Get a custom crash report metadata field that has been set.
SetUserMetadata	Set a custom metadata key-value pair to be included with crash reports.

Description

A zero argument persistent callback that can be saved with the Scene.

```
using UnityEngine;
using UnityEngine.Events;
using System.Collections;


public class ExampleClass : MonoBehaviour
{
    UnityEvent m_MyEvent;


    void Start()
    {
        if (m_MyEvent == null)
            m_MyEvent = new UnityEvent();


        m_MyEvent.AddListener(Ping);
    }


    void Update()
    {
        if (Input.anyKeyDown && m_MyEvent != null)
        {
            m_MyEvent.Invoke();
        }
    }


    void Ping()
    {
        Debug.Log("Ping");
    }
}
```

Constructors

UnityEvent	Constructor.

Public Methods

AddListener	Add a non persistent listener to the UnityEvent.
Invoke	Invoke all registered callbacks (runtime and persistent).
RemoveListener	Remove a non persistent listener from the UnityEvent.

Inherited Members

Public Methods

GetPersistentEventCount	Get the number of registered persistent listeners.
GetPersistentMethodName	Get the target method name of the listener at index index.
GetPersistentTarget	Get the target component of the listener at index index.
RemoveAllListeners	Remove all non-persisent (ie created from script) listeners from the event.
SetPersistentListenerState	Modify the execution state of a persistent listener.

Static Methods

GetValidMethodInfo	Given an object, function name, and a list of argument types; find the method that matches.

Description

One argument version of UnityEvent.

If you wish to use a generic UnityEvent type you must override the class type.

```
using UnityEngine;
using UnityEngine.Events;


[System.Serializable]
public class MyIntEvent : UnityEvent<int>
{
}


public class ExampleClass : MonoBehaviour
{
    public MyIntEvent m_MyEvent;


    void Start()
    {
        if (m_MyEvent == null)
            m_MyEvent = new MyIntEvent();


        m_MyEvent.AddListener(Ping);
    }


    void Update()
    {
        if (Input.anyKeyDown && m_MyEvent != null)
        {
            m_MyEvent.Invoke(5);
        }
    }


    void Ping(int i)
    {
        Debug.Log("Ping" + i);
```

```
    }
}
```

Inherited Members

Public Methods

GetPersistentEventCount	Get the number of registered persistent listeners.
GetPersistentMethodName	Get the target method name of the listener at index index.
GetPersistentTarget	Get the target component of the listener at index index.
RemoveAllListeners	Remove all non-persisent (ie created from script) listeners from the event.
SetPersistentListenerState	Modify the execution state of a persistent listener.

Static Methods

GetValidMethodInfo	Given an object, function name, and a list of argument types; find the method that matches.

Description

Two argument version of UnityEvent.

If you wish to use a generic UnityEvent type you must override the class type.

```
using UnityEngine;
using UnityEngine.Events;


[System.Serializable]
public class MyIntEvent : UnityEvent<int, int>
{
}


public class ExampleClass : MonoBehaviour
{
    public MyIntEvent m_MyEvent;


    void Start()
    {
        if (m_MyEvent == null)
            m_MyEvent = new MyIntEvent();


        m_MyEvent.AddListener(Ping);
    }


    void Update()
    {
        if (Input.anyKeyDown && m_MyEvent != null)
        {
            m_MyEvent.Invoke(5, 6);
        }
    }


    void Ping(int i, int j)
    {
        Debug.Log("Ping" + i + ", " + j);
```

```
    }
}
```

Inherited Members

Public Methods

GetPersistentEventCount	Get the number of registered persistent listeners.
GetPersistentMethodName	Get the target method name of the listener at index index.
GetPersistentTarget	Get the target component of the listener at index index.
RemoveAllListeners	Remove all non-persisent (ie created from script) listeners from the event.
SetPersistentListenerState	Modify the execution state of a persistent listener.

Static Methods

GetValidMethodInfo	Given an object, function name, and a list of argument types; find the method that matches.

Description

Three argument version of UnityEvent.

If you wish to use a generic UnityEvent type you must override the class type.

```
using UnityEngine;
using UnityEngine.Events;


[System.Serializable]
public class MyIntEvent : UnityEvent<int, int, int>
{
}


public class ExampleClass : MonoBehaviour
{
    public MyIntEvent m_MyEvent;


    void Start()
    {
        if (m_MyEvent == null)
            m_MyEvent = new MyIntEvent();


        m_MyEvent.AddListener(Ping);
    }


    void Update()
    {
        if (Input.anyKeyDown && m_MyEvent != null)
        {
            m_MyEvent.Invoke(5, 6, 7);
        }
    }


    void Ping(int i, int j, int k)
    {
        Debug.Log("Ping" + i + j + k);
```

```
    }
}
```

Inherited Members

Public Methods

GetPersistentEventCount	Get the number of registered persistent listeners.
GetPersistentMethodName	Get the target method name of the listener at index index.
GetPersistentTarget	Get the target component of the listener at index index.
RemoveAllListeners	Remove all non-persisent (ie created from script) listeners from the event.
SetPersistentListenerState	Modify the execution state of a persistent listener.

Static Methods

GetValidMethodInfo	Given an object, function name, and a list of argument types; find the method that matches.

Description

Four argument version of UnityEvent.

If you wish to use a generic UnityEvent type you must override the class type.

```
using UnityEngine;
using UnityEngine.Events;



[System.Serializable]
public class MyIntEvent : UnityEvent<int, int, int, int>
{
}


public class ExampleClass : MonoBehaviour
{
    public MyIntEvent m_MyEvent;


    void Start()
    {
        if (m_MyEvent == null)
            m_MyEvent = new MyIntEvent();


        m_MyEvent.AddListener(Ping);
    }


    void Update()
    {
        if (Input.anyKeyDown && m_MyEvent != null)
        {
            m_MyEvent.Invoke(5, 6, 7, 8);
        }
    }


    void Ping(int i, int j, int k, int l)
    {
```

```
        Debug.Log("Ping" + i + j + k + l);
    }
}
```

Inherited Members

Public Methods

GetPersistentEventCount	Get the number of registered persistent listeners.
GetPersistentMethodName	Get the target method name of the listener at index index.
GetPersistentTarget	Get the target component of the listener at index index.
RemoveAllListeners	Remove all non-persisent (ie created from script) listeners from the event.
SetPersistentListenerState	Modify the execution state of a persistent listener.

Static Methods

GetValidMethodInfo	Given an object, function name, and a list of argument types; find the method that matches.

Implements interfaces:ISerializationCallbackReceiver

Description

Abstract base class for UnityEvents.

This class provides the base functionality for the UnityEvents.

Public Methods

GetPersistentEventCount	Get the number of registered persistent listeners.
GetPersistentMethodName	Get the target method name of the listener at index index.
GetPersistentTarget	Get the target component of the listener at index index.
RemoveAllListeners	Remove all non-persisent (ie created from script) listeners from the event.
SetPersistentListenerState	Modify the execution state of a persistent listener.

Static Methods

GetValidMethodInfo	Given an object, function name, and a list of argument types; find the method that matches.

Description

THe mode that a listener is operating in.

Properties

EventDefined	The listener will use the function binding specified by the even.
Void	The listener will bind to zero argument functions.
Object	The listener will bind to one argument Object functions.
Int	The listener will bind to one argument int functions.
Float	The listener will bind to one argument float functions.
String	The listener will bind to one argument string functions.
Bool	The listener will bind to one argument bool functions.

Description

Controls the scope of UnityEvent callbacks.

Properties

Off	Callback is not issued.
EditorAndRuntime	Callback is always issued.
RuntimeOnly	Callback is only issued in the Runtime and Editor playmode.

Description

A class that can be used for sending simple events via the event system.

Properties

used	Is the event used?

Public Methods

Reset	Reset the event.
Use	Use the event.

Description

Event Data associated with Axis Events (Controller / Keyboard).

See Also: BaseEventData.

Properties

moveDir	MoveDirection for this event.
moveVector	Raw input vector associated with this event.

Inherited Members

Properties

used	Is the event used?
currentInputModule	A reference to the BaseInputModule that sent this event.
selectedObject	The object currently considered selected by the EventSystem.

Public Methods

Reset	Reset the event.
Use	Use the event.

Description

A class that contains the base event data that is common to all event types in the new EventSystem.

Properties

currentInputModule	A reference to the BaseInputModule that sent this event.
selectedObject	The object currently considered selected by the EventSystem.

Constructors

BaseEventData	Construct a BaseEventData tied to the passed EventSystem.

Inherited Members

Properties

used	Is the event used?

Public Methods

Reset	Reset the event.
Use	Use the event.

Description

Interface to the Input system used by the BaseInputModule. With this it is possible to bypass the Input system with your own but still use the same InputModule. For example this can be used to feed fake input into the UI or interface with a different input system.

Properties

compositionCursorPos	Interface to Input.compositionCursorPos. Can be overridden to provide custom input instead of using the Input class.
compositionString	Interface to Input.compositionString. Can be overridden to provide custom input instead of using the Input class.
imeCompositionMode	Interface to Input.imeCompositionMode. Can be overridden to provide custom input instead of using the Input class.
mousePosition	Interface to Input.mousePosition. Can be overridden to provide custom input instead of using the Input class.
mousePresent	Interface to Input.mousePresent. Can be overridden to provide custom input instead of using the Input class.
mouseScrollDelta	Interface to Input.mouseScrollDelta. Can be overridden to provide custom input instead of using the Input class.
touchCount	Interface to Input.touchCount. Can be overridden to provide custom input instead of using the Input class.
touchSupported	Interface to Input.touchSupported. Can be overridden to provide custom input instead of using the Input class.

Public Methods

GetAxisRaw	Interface to Input.GetAxisRaw. Can be overridden to provide custom input instead of using the Input class.
GetButtonDown	Interface to Input.GetButtonDown. Can be overridden to provide custom input instead of using the Input class.
GetMouseButton	Interface to Input.GetMouseButton. Can be overridden to provide custom input instead of using the Input class.
GetMouseButtonDown	Interface to Input.GetMouseButtonDown. Can be overridden to provide custom input instead of using the Input class.

GetMouseButtonUp	Interface to Input.GetMouseButtonUp. Can be overridden to provide custom input instead of using the Input class.
GetTouch	Interface to Input.GetTouch. Can be overridden to provide custom input instead of using the Input class.

Inherited Members

Properties

enabled	Enabled Behaviours are Updated, disabled Behaviours are not.
isActiveAndEnabled	Has the Behaviour had active and enabled called?
gameObject	The game object this component is attached to. A component is always attached to a game object.
tag	The tag of this game object.
transform	The Transform attached to this GameObject.
runInEditMode	Allow a specific instance of a MonoBehaviour to run in edit mode (only available in the editor).
useGUILayout	Disabling this lets you skip the GUI layout phase.
hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.

Public Methods

BroadcastMessage	Calls the method named methodName on every MonoBehaviour in this game object or any of its children.
CompareTag	Is this game object tagged with tag ?
GetComponent	Returns the component of Type type if the game object has one attached, null if it doesn't.
GetComponentInChildren	Returns the component of Type type in the GameObject or any of its children using depth first search.
GetComponentInParent	Returns the component of Type type in the GameObject or any of its parents.
GetComponents	Returns all components of Type type in the GameObject.
GetComponentsInChildren	Returns all components of Type type in the GameObject or any of its children.
GetComponentsInParent	Returns all components of Type type in the GameObject or any of its parents.

SendMessage	Calls the method named methodName on every MonoBehaviour in this game object.
SendMessageUpwards	Calls the method named methodName on every MonoBehaviour in this game object and on every ancestor of the behaviour.
CancelInvoke	Cancels all Invoke calls on this MonoBehaviour.
Invoke	Invokes the method methodName in time seconds.
InvokeRepeating	Invokes the method methodName in time seconds, then repeatedly every repeatRate seconds.
IsInvoking	Is any invoke on methodName pending?
StartCoroutine	Starts a coroutine.
StopAllCoroutines	Stops all coroutines running on this behaviour.
StopCoroutine	Stops the first coroutine named methodName, or the coroutine stored in routine running on this behaviour.
GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.
IsActive	Returns true if the GameObject and the Component are active.
IsDestroyed	Returns true if the native representation of the behaviour has been destroyed.

Protected Methods

Awake	See MonoBehaviour.Awake.
OnBeforeTransformParentChanged	See MonoBehaviour.OnBeforeTransformParentChanged.
OnCanvasGroupChanged	See MonoBehaviour.OnCanvasGroupChanged.
OnCanvasHierarchyChanged	Called when the state of the parent Canvas is changed.
OnDestroy	See MonoBehaviour.OnDestroy.
OnDidApplyAnimationProperties	See LayoutGroup.OnDidApplyAnimationProperties.
OnDisable	See MonoBehaviour.OnDisable.
OnEnable	See MonoBehaviour.OnEnable.
OnRectTransformDimensionsChange	This callback is called if an associated RectTransform has its dimensions changed.
OnTransformParentChanged	See MonoBehaviour.OnRectTransformParentChanged.

OnValidate	See MonoBehaviour.OnValidate.
Reset	See MonoBehaviour.Reset.
Start	See MonoBehaviour.Start.

Static Methods

print	Logs message to the Unity Console (identical to Debug.Log).
Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Messages

Awake	Awake is called when the script instance is being loaded.
FixedUpdate	This function is called every fixed framerate frame, if the MonoBehaviour is enabled.
LateUpdate	LateUpdate is called every frame, if the Behaviour is enabled.
OnAnimatorIK	Callback for setting up animation IK (inverse kinematics).
OnAnimatorMove	Callback for processing animation movements for modifying root motion.
OnApplicationFocus	Sent to all GameObjects when the player gets or loses focus.
OnApplicationPause	Sent to all GameObjects when the application pauses.
OnApplicationQuit	Sent to all game objects before the application quits.

OnAudioFilterRead	If OnAudioFilterRead is implemented, Unity will insert a custom filter into the audio DSP chain.
OnBecameInvisible	OnBecameInvisible is called when the renderer is no longer visible by any camera.
OnBecameVisible	OnBecameVisible is called when the renderer became visible by any camera.
OnCollisionEnter	OnCollisionEnter is called when this collider/rigidbody has begun touching another rigidbody/collider.
OnCollisionEnter2D	Sent when an incoming collider makes contact with this object's collider (2D physics only).
OnCollisionExit	OnCollisionExit is called when this collider/rigidbody has stopped touching another rigidbody/collider.
OnCollisionExit2D	Sent when a collider on another object stops touching this object's collider (2D physics only).
OnCollisionStay	:ref::OnCollisionStay is called once per frame for every collider/rigidbody that is touching rigidbody/collider.
OnCollisionStay2D	Sent each frame where a collider on another object is touching this object's collider (2D physics only).
OnConnectedToServer	Called on the client when you have successfully connected to a server.
OnControllerColliderHit	OnControllerColliderHit is called when the controller hits a collider while performing a Move.
OnDestroy	Destroying the attached Behaviour will result in the game or Scene receiving OnDestroy.
OnDisable	This function is called when the behaviour becomes disabled.
OnDisconnectedFromServer	Called on the client when the connection was lost or you disconnected from the server.
OnDrawGizmos	Implement OnDrawGizmos if you want to draw gizmos that are also pickable and always drawn.
OnDrawGizmosSelected	Implement OnDrawGizmosSelected to draw a gizmo if the object is selected.
OnEnable	This function is called when the object becomes enabled and active.
OnFailedToConnect	Called on the client when a connection attempt fails for some reason.
OnFailedToConnectToMasterServer	Called on clients or servers when there is a problem connecting to the MasterServer.

OnGUI	OnGUI is called for rendering and handling GUI events.
OnJointBreak	Called when a joint attached to the same game object broke.
OnJointBreak2D	Called when a Joint2D attached to the same game object breaks.
OnMasterServerEvent	Called on clients or servers when reporting events from the MasterServer.
OnMouseDown	OnMouseDown is called when the user has pressed the mouse button while over the GUIElement or Collider.
OnMouseDrag	OnMouseDrag is called when the user has clicked on a GUIElement or Collider and is still holding down the mouse.
OnMouseEnter	Called when the mouse enters the GUIElement or Collider.
OnMouseExit	Called when the mouse is not any longer over the GUIElement or Collider.
OnMouseOver	Called every frame while the mouse is over the GUIElement or Collider.
OnMouseUp	OnMouseUp is called when the user has released the mouse button.
OnMouseUpAsButton	OnMouseUpAsButton is only called when the mouse is released over the same GUIElement or Collider as it was pressed.
OnNetworkInstantiate	Called on objects which have been network instantiated with Network.Instantiate.
OnParticleCollision	OnParticleCollision is called when a particle hits a Collider.
OnParticleSystemStopped	OnParticleSystemStopped is called when all particles in the system have died, and no new particles will be born. New particles cease to be created either after Stop is called, or when the duration property of a non-looping system has been exceeded.
OnParticleTrigger	OnParticleTrigger is called when any particles in a particle system meet the conditions in the trigger module.
OnPlayerConnected	Called on the server whenever a new player has successfully connected.
OnPlayerDisconnected	Called on the server whenever a player disconnected from the server.
OnPostRender	OnPostRender is called after a camera finished rendering the Scene.
OnPreCull	OnPreCull is called before a camera culls the Scene.
OnPreRender	OnPreRender is called before a camera starts rendering the Scene.
OnRenderImage	OnRenderImage is called after all rendering is complete to render image.

OnRenderObject	OnRenderObject is called after camera has rendered the Scene.
OnSerializeNetworkView	Used to customize synchronization of variables in a script watched by a network view.
OnServerInitialized	Called on the server whenever a Network.InitializeServer was invoked and has completed.
OnTransformChildrenChanged	This function is called when the list of children of the transform of the GameObject has changed.
OnTransformParentChanged	This function is called when the parent property of the transform of the GameObject has changed.
OnTriggerEnter	OnTriggerEnter is called when the Collider other enters the trigger.
OnTriggerEnter2D	Sent when another object enters a trigger collider attached to this object (2D physics only).
OnTriggerExit	OnTriggerExit is called when the Collider other has stopped touching the trigger.
OnTriggerExit2D	Sent when another object leaves a trigger collider attached to this object (2D physics only).
OnTriggerStay	OnTriggerStay is called once per physics update for every Collider other that is touching the trigger.
OnTriggerStay2D	Sent each frame where another object is within a trigger collider attached to this object (2D physics only).
OnValidate	This function is called when the script is loaded or a value is changed in the inspector (Called in the editor only).
OnWillRenderObject	OnWillRenderObject is called for each camera if the object is visible and not a UI element.
Reset	Reset to default values.
Start	Start is called on the frame when a script is enabled just before any of the Update methods are called the first time.
Update	Update is called every frame, if the MonoBehaviour is enabled.

Description

A base module that raises events and sends them to GameObjects.

An Input Module is a component of the EventSystem that is responsible for raising events and sending them to GameObjects for handling. The BaseInputModule is a class that all Input Modules in the EventSystem inherit from. Examples of provided modules are TouchInputModule and StandaloneInputModule, if these are inadequate for your project you can create your own by extending from the BaseInputModule.

```
using UnityEngine;
using UnityEngine.EventSystems;


// Create a module that every tick sends a 'Move' event to
// the target object


public class MyInputModule : BaseInputModule
{
    public GameObject m_TargetObject;


    public override void Process()
    {
        if (m_TargetObject == null)
            return;
        ExecuteEvents.Execute(m_TargetObject, new BaseEventData(eventSystem), Ex
    }
}
```

Properties

input	The current BaseInput being used by the input module.
inputOverride	Used to override the default BaseInput for the input module.

Public Methods

ActivateModule	Called when the module is activated. Override this if you want custom code to execute when you activate your module.

DeactivateModule	Called when the module is deactivated. Override this if you want custom code to execute when you deactivate your module.
IsModuleSupported	Check to see if the module is supported. Override this if you have a platfrom specific module (eg. TouchInputModule that you do not want to activate on standalone.
IsPointerOverGameObject	Is the pointer with the given ID over an EventSystem object?
Process	Process the current tick for the module.
ShouldActivateModule	Should be activated.
UpdateModule	Update the internal state of the Module.

Protected Methods

GetAxisEventData	Given some input data generate an AxisEventData that can be used by the event system.
GetBaseEventData	Generate a BaseEventData that can be used by the EventSystem.
HandlePointerExitAndEnter	Handle sending enter and exit events when a new enter target is found.
OnDisable	See MonoBehaviour.OnDisable.
OnEnable	See MonoBehaviour.OnEnable.

Static Methods

DetermineMoveDirection	Given an input movement, determine the best MoveDirection.
FindCommonRoot	Given 2 GameObjects, return a common root GameObject (or null).
FindFirstRaycast	Return the first valid RaycastResult.

Inherited Members

Properties

enabled	Enabled Behaviours are Updated, disabled Behaviours are not.
isActiveAndEnabled	Has the Behaviour had active and enabled called?
gameObject	The game object this component is attached to. A component is always attached to a game object.
tag	The tag of this game object.

transform	The Transform attached to this GameObject.
runInEditMode	Allow a specific instance of a MonoBehaviour to run in edit mode (only available in the editor).
useGUILayout	Disabling this lets you skip the GUI layout phase.
hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.

Public Methods

BroadcastMessage	Calls the method named methodName on every MonoBehaviour in this game object or any of its children.
CompareTag	Is this game object tagged with tag ?
GetComponent	Returns the component of Type type if the game object has one attached, null if it doesn't.
GetComponentInChildren	Returns the component of Type type in the GameObject or any of its children using depth first search.
GetComponentInParent	Returns the component of Type type in the GameObject or any of its parents.
GetComponents	Returns all components of Type type in the GameObject.
GetComponentsInChildren	Returns all components of Type type in the GameObject or any of its children.
GetComponentsInParent	Returns all components of Type type in the GameObject or any of its parents.
SendMessage	Calls the method named methodName on every MonoBehaviour in this game object.
SendMessageUpwards	Calls the method named methodName on every MonoBehaviour in this game object and on every ancestor of the behaviour.
CancelInvoke	Cancels all Invoke calls on this MonoBehaviour.
Invoke	Invokes the method methodName in time seconds.
InvokeRepeating	Invokes the method methodName in time seconds, then repeatedly every repeatRate seconds.
IsInvoking	Is any invoke on methodName pending?
StartCoroutine	Starts a coroutine.
StopAllCoroutines	Stops all coroutines running on this behaviour.
StopCoroutine	Stops the first coroutine named methodName, or the coroutine stored in routine

	running on this behaviour.
GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.
IsActive	Returns true if the GameObject and the Component are active.
IsDestroyed	Returns true if the native representation of the behaviour has been destroyed.

Protected Methods

Awake	See MonoBehaviour.Awake.
OnBeforeTransformParentChanged	See MonoBehaviour.OnBeforeTransformParentChanged.
OnCanvasGroupChanged	See MonoBehaviour.OnCanvasGroupChanged.
OnCanvasHierarchyChanged	Called when the state of the parent Canvas is changed.
OnDestroy	See MonoBehaviour.OnDestroy.
OnDidApplyAnimationProperties	See LayoutGroup.OnDidApplyAnimationProperties.
OnRectTransformDimensionsChange	This callback is called if an associated RectTransform has its dimensions changed.
OnTransformParentChanged	See MonoBehaviour.OnRectTransformParentChanged.
OnValidate	See MonoBehaviour.OnValidate.
Reset	See MonoBehaviour.Reset.
Start	See MonoBehaviour.Start.

Static Methods

print	Logs message to the Unity Console (identical to Debug.Log).
Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Messages

Awake	Awake is called when the script instance is being loaded.
FixedUpdate	This function is called every fixed framerate frame, if the MonoBehaviour is enabled.
LateUpdate	LateUpdate is called every frame, if the Behaviour is enabled.
OnAnimatorIK	Callback for setting up animation IK (inverse kinematics).
OnAnimatorMove	Callback for processing animation movements for modifying root motion.
OnApplicationFocus	Sent to all GameObjects when the player gets or loses focus.
OnApplicationPause	Sent to all GameObjects when the application pauses.
OnApplicationQuit	Sent to all game objects before the application quits.
OnAudioFilterRead	If OnAudioFilterRead is implemented, Unity will insert a custom filter into the audio DSP chain.
OnBecameInvisible	OnBecameInvisible is called when the renderer is no longer visible by any camera.
OnBecameVisible	OnBecameVisible is called when the renderer became visible by any camera.
OnCollisionEnter	OnCollisionEnter is called when this collider/rigidbody has begun touching another rigidbody/collider.
OnCollisionEnter2D	Sent when an incoming collider makes contact with this object's collider (2D physics only).
OnCollisionExit	OnCollisionExit is called when this collider/rigidbody has stopped touching another rigidbody/collider.
OnCollisionExit2D	Sent when a collider on another object stops touching this object's collider (2D physics only).
OnCollisionStay	:ref::OnCollisionStay is called once per frame for every collider/rigidbody that is touching rigidbody/collider.

OnCollisionStay2D	Sent each frame where a collider on another object is touching this object's collider (2D physics only).
OnConnectedToServer	Called on the client when you have successfully connected to a server.
OnControllerColliderHit	OnControllerColliderHit is called when the controller hits a collider while performing a Move.
OnDestroy	Destroying the attached Behaviour will result in the game or Scene receiving OnDestroy.
OnDisconnectedFromServer	Called on the client when the connection was lost or you disconnected from the server.
OnDrawGizmos	Implement OnDrawGizmos if you want to draw gizmos that are also pickable and always drawn.
OnDrawGizmosSelected	Implement OnDrawGizmosSelected to draw a gizmo if the object is selected.
OnFailedToConnect	Called on the client when a connection attempt fails for some reason.
OnFailedToConnectToMasterServer	Called on clients or servers when there is a problem connecting to the MasterServer.
OnGUI	OnGUI is called for rendering and handling GUI events.
OnJointBreak	Called when a joint attached to the same game object broke.
OnJointBreak2D	Called when a Joint2D attached to the same game object breaks.
OnMasterServerEvent	Called on clients or servers when reporting events from the MasterServer.
OnMouseDown	OnMouseDown is called when the user has pressed the mouse button while over the GUIElement or Collider.
OnMouseDrag	OnMouseDrag is called when the user has clicked on a GUIElement or Collider and is still holding down the mouse.
OnMouseEnter	Called when the mouse enters the GUIElement or Collider.
OnMouseExit	Called when the mouse is not any longer over the GUIElement or Collider.
OnMouseOver	Called every frame while the mouse is over the GUIElement or Collider.
OnMouseUp	OnMouseUp is called when the user has released the mouse button.
OnMouseUpAsButton	OnMouseUpAsButton is only called when the mouse is released over the same GUIElement or Collider as it was pressed.
OnNetworkInstantiate	Called on objects which have been network instantiated with

	Network.Instantiate.
OnParticleCollision	OnParticleCollision is called when a particle hits a Collider.
OnParticleSystemStopped	OnParticleSystemStopped is called when all particles in the system have died, and no new particles will be born. New particles cease to be created either after Stop is called, or when the duration property of a non-looping system has been exceeded.
OnParticleTrigger	OnParticleTrigger is called when any particles in a particle system meet the conditions in the trigger module.
OnPlayerConnected	Called on the server whenever a new player has successfully connected.
OnPlayerDisconnected	Called on the server whenever a player disconnected from the server.
OnPostRender	OnPostRender is called after a camera finished rendering the Scene.
OnPreCull	OnPreCull is called before a camera culls the Scene.
OnPreRender	OnPreRender is called before a camera starts rendering the Scene.
OnRenderImage	OnRenderImage is called after all rendering is complete to render image.
OnRenderObject	OnRenderObject is called after camera has rendered the Scene.
OnSerializeNetworkView	Used to customize synchronization of variables in a script watched by a network view.
OnServerInitialized	Called on the server whenever a Network.InitializeServer was invoked and has completed.
OnTransformChildrenChanged	This function is called when the list of children of the transform of the GameObject has changed.
OnTransformParentChanged	This function is called when the parent property of the transform of the GameObject has changed.
OnTriggerEnter	OnTriggerEnter is called when the Collider other enters the trigger.
OnTriggerEnter2D	Sent when another object enters a trigger collider attached to this object (2D physics only).
OnTriggerExit	OnTriggerExit is called when the Collider other has stopped touching the trigger.
OnTriggerExit2D	Sent when another object leaves a trigger collider attached to this object (2D physics only).
OnTriggerStay	OnTriggerStay is called once per physics update for every Collider other that is touching the trigger.

OnTriggerStay2D	Sent each frame where another object is within a trigger collider attached to this object (2D physics only).
OnValidate	This function is called when the script is loaded or a value is changed in the inspector (Called in the editor only).
OnWillRenderObject	OnWillRenderObject is called for each camera if the object is visible and not a UI element.
Reset	Reset to default values.
Start	Start is called on the frame when a script is enabled just before any of the Update methods are called the first time.
Update	Update is called every frame, if the MonoBehaviour is enabled.

Description

Base class for any RayCaster.

A Raycaster is responsible for raycasting against Scene elements to determine if the cursor is over them. Default Raycasters include PhysicsRaycaster, Physics2DRaycaster, GraphicRaycaster.

Custom raycasters can be added by extending this class.

Properties

eventCamera	The camera that will generate rays for this raycaster.
renderOrderPriority	Priority of the raycaster based upon render order.
sortOrderPriority	Priority of the raycaster based upon sort order.

Public Methods

Raycast	Raycast against the Scene.

Protected Methods

OnDisable	See MonoBehaviour.OnDisable.

Inherited Members

Properties

enabled	Enabled Behaviours are Updated, disabled Behaviours are not.
isActiveAndEnabled	Has the Behaviour had active and enabled called?
gameObject	The game object this component is attached to. A component is always attached to a game object.
tag	The tag of this game object.
transform	The Transform attached to this GameObject.
runInEditMode	Allow a specific instance of a MonoBehaviour to run in edit mode (only available in the editor).
useGUILayout	Disabling this lets you skip the GUI layout phase.

hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.

Public Methods

BroadcastMessage	Calls the method named methodName on every MonoBehaviour in this game object or any of its children.
CompareTag	Is this game object tagged with tag ?
GetComponent	Returns the component of Type type if the game object has one attached, null if it doesn't.
GetComponentInChildren	Returns the component of Type type in the GameObject or any of its children using depth first search.
GetComponentInParent	Returns the component of Type type in the GameObject or any of its parents.
GetComponents	Returns all components of Type type in the GameObject.
GetComponentsInChildren	Returns all components of Type type in the GameObject or any of its children.
GetComponentsInParent	Returns all components of Type type in the GameObject or any of its parents.
SendMessage	Calls the method named methodName on every MonoBehaviour in this game object.
SendMessageUpwards	Calls the method named methodName on every MonoBehaviour in this game object and on every ancestor of the behaviour.
CancelInvoke	Cancels all Invoke calls on this MonoBehaviour.
Invoke	Invokes the method methodName in time seconds.
InvokeRepeating	Invokes the method methodName in time seconds, then repeatedly every repeatRate seconds.
IsInvoking	Is any invoke on methodName pending?
StartCoroutine	Starts a coroutine.
StopAllCoroutines	Stops all coroutines running on this behaviour.
StopCoroutine	Stops the first coroutine named methodName, or the coroutine stored in routine running on this behaviour.
GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.
IsActive	Returns true if the GameObject and the Component are active.

IsDestroyed	Returns true if the native representation of the behaviour has been destroyed.

Protected Methods

Awake	See MonoBehaviour.Awake.
OnBeforeTransformParentChanged	See MonoBehaviour.OnBeforeTransformParentChanged.
OnCanvasGroupChanged	See MonoBehaviour.OnCanvasGroupChanged.
OnCanvasHierarchyChanged	Called when the state of the parent Canvas is changed.
OnDestroy	See MonoBehaviour.OnDestroy.
OnDidApplyAnimationProperties	See LayoutGroup.OnDidApplyAnimationProperties.
OnEnable	See MonoBehaviour.OnEnable.
OnRectTransformDimensionsChange	This callback is called if an associated RectTransform has its dimensions changed.
OnTransformParentChanged	See MonoBehaviour.OnRectTransformParentChanged.
OnValidate	See MonoBehaviour.OnValidate.
Reset	See MonoBehaviour.Reset.
Start	See MonoBehaviour.Start.

Static Methods

print	Logs message to the Unity Console (identical to Debug.Log).
Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.

Operators

bool	Does the object exist?

operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Messages

Awake	Awake is called when the script instance is being loaded.
FixedUpdate	This function is called every fixed framerate frame, if the MonoBehaviour is enabled.
LateUpdate	LateUpdate is called every frame, if the Behaviour is enabled.
OnAnimatorIK	Callback for setting up animation IK (inverse kinematics).
OnAnimatorMove	Callback for processing animation movements for modifying root motion.
OnApplicationFocus	Sent to all GameObjects when the player gets or loses focus.
OnApplicationPause	Sent to all GameObjects when the application pauses.
OnApplicationQuit	Sent to all game objects before the application quits.
OnAudioFilterRead	If OnAudioFilterRead is implemented, Unity will insert a custom filter into the audio DSP chain.
OnBecameInvisible	OnBecameInvisible is called when the renderer is no longer visible by any camera.
OnBecameVisible	OnBecameVisible is called when the renderer became visible by any camera.
OnCollisionEnter	OnCollisionEnter is called when this collider/rigidbody has begun touching another rigidbody/collider.
OnCollisionEnter2D	Sent when an incoming collider makes contact with this object's collider (2D physics only).
OnCollisionExit	OnCollisionExit is called when this collider/rigidbody has stopped touching another rigidbody/collider.
OnCollisionExit2D	Sent when a collider on another object stops touching this object's collider (2D physics only).
OnCollisionStay	:ref::OnCollisionStay is called once per frame for every collider/rigidbody that is touching rigidbody/collider.
OnCollisionStay2D	Sent each frame where a collider on another object is touching this object's collider (2D physics only).
OnConnectedToServer	Called on the client when you have successfully connected to a server.

OnControllerColliderHit	OnControllerColliderHit is called when the controller hits a collider while performing a Move.
OnDestroy	Destroying the attached Behaviour will result in the game or Scene receiving OnDestroy.
OnDisconnectedFromServer	Called on the client when the connection was lost or you disconnected from the server.
OnDrawGizmos	Implement OnDrawGizmos if you want to draw gizmos that are also pickable and always drawn.
OnDrawGizmosSelected	Implement OnDrawGizmosSelected to draw a gizmo if the object is selected.
OnEnable	This function is called when the object becomes enabled and active.
OnFailedToConnect	Called on the client when a connection attempt fails for some reason.
OnFailedToConnectToMasterServer	Called on clients or servers when there is a problem connecting to the MasterServer.
OnGUI	OnGUI is called for rendering and handling GUI events.
OnJointBreak	Called when a joint attached to the same game object broke.
OnJointBreak2D	Called when a Joint2D attached to the same game object breaks.
OnMasterServerEvent	Called on clients or servers when reporting events from the MasterServer.
OnMouseDown	OnMouseDown is called when the user has pressed the mouse button while over the GUIElement or Collider.
OnMouseDrag	OnMouseDrag is called when the user has clicked on a GUIElement or Collider and is still holding down the mouse.
OnMouseEnter	Called when the mouse enters the GUIElement or Collider.
OnMouseExit	Called when the mouse is not any longer over the GUIElement or Collider.
OnMouseOver	Called every frame while the mouse is over the GUIElement or Collider.
OnMouseUp	OnMouseUp is called when the user has released the mouse button.
OnMouseUpAsButton	OnMouseUpAsButton is only called when the mouse is released over the same GUIElement or Collider as it was pressed.
OnNetworkInstantiate	Called on objects which have been network instantiated with Network.Instantiate.

OnParticleCollision	OnParticleCollision is called when a particle hits a Collider.
OnParticleSystemStopped	OnParticleSystemStopped is called when all particles in the system have died, and no new particles will be born. New particles cease to be created either after Stop is called, or when the duration property of a non-looping system has been exceeded.
OnParticleTrigger	OnParticleTrigger is called when any particles in a particle system meet the conditions in the trigger module.
OnPlayerConnected	Called on the server whenever a new player has successfully connected.
OnPlayerDisconnected	Called on the server whenever a player disconnected from the server.
OnPostRender	OnPostRender is called after a camera finished rendering the Scene.
OnPreCull	OnPreCull is called before a camera culls the Scene.
OnPreRender	OnPreRender is called before a camera starts rendering the Scene.
OnRenderImage	OnRenderImage is called after all rendering is complete to render image.
OnRenderObject	OnRenderObject is called after camera has rendered the Scene.
OnSerializeNetworkView	Used to customize synchronization of variables in a script watched by a network view.
OnServerInitialized	Called on the server whenever a Network.InitializeServer was invoked and has completed.
OnTransformChildrenChanged	This function is called when the list of children of the transform of the GameObject has changed.
OnTransformParentChanged	This function is called when the parent property of the transform of the GameObject has changed.
OnTriggerEnter	OnTriggerEnter is called when the Collider other enters the trigger.
OnTriggerEnter2D	Sent when another object enters a trigger collider attached to this object (2D physics only).
OnTriggerExit	OnTriggerExit is called when the Collider other has stopped touching the trigger.
OnTriggerExit2D	Sent when another object leaves a trigger collider attached to this object (2D physics only).
OnTriggerStay	OnTriggerStay is called once per physics update for every Collider other that is touching the trigger.
OnTriggerStay2D	Sent each frame where another object is within a trigger collider attached to this object (2D physics only).

OnValidate	This function is called when the script is loaded or a value is changed in the inspector (Called in the editor only).
OnWillRenderObject	OnWillRenderObject is called for each camera if the object is visible and not a UI element.
Reset	Reset to default values.
Start	Start is called on the frame when a script is enabled just before any of the Update methods are called the first time.
Update	Update is called every frame, if the MonoBehaviour is enabled.

Description

Handles input, raycasting, and sending events.

The EventSystem is responsible for processing and handling events in a Unity Scene. A Scene should only contain one EventSystem. The EventSystem works in conjunction with a number of modules and mostly just holds state and delegates functionality to specific, overrideable components.

When the EventSystem is started it searches for any BaseInputModules attached to the same GameObject and adds them to an internal list. On update each attached module receives an UpdateModules call, where the module can modify internal state. After each module has been Updated the active module has the Process call executed. This is where custom module processing can take place.

Static Properties

current	Return the current EventSystem.

Properties

alreadySelecting	Returns true if the EventSystem is already in a SetSelectedGameObject.
currentInputModule	The currently active BaseInputModule.
currentSelectedGameObject	The GameObject currently considered active by the EventSystem.
firstSelectedGameObject	The GameObject that was selected first.
isFocused	Flag to say whether the EventSystem thinks it should be paused or not based upon focused state.
pixelDragThreshold	The soft area for dragging in pixels.
sendNavigationEvents	Should the EventSystem allow navigation events (move / submit / cancel).

Public Methods

IsPointerOverGameObject	Is the pointer with the given ID over an EventSystem object?
RaycastAll	Raycast into the Scene using all configured BaseRaycasters.
SetSelectedGameObject	Set the GameObject as selected. Will send an OnDeselect the the old selected object and OnSelect to the new selected object.
UpdateModules	Recalculate the internal list of BaseInputModules.

Protected Methods

OnDisable	See MonoBehaviour.OnDisable.

Inherited Members

Properties

enabled	Enabled Behaviours are Updated, disabled Behaviours are not.
isActiveAndEnabled	Has the Behaviour had active and enabled called?
gameObject	The game object this component is attached to. A component is always attached to a game object.
tag	The tag of this game object.
transform	The Transform attached to this GameObject.
runInEditMode	Allow a specific instance of a MonoBehaviour to run in edit mode (only available in the editor).
useGUILayout	Disabling this lets you skip the GUI layout phase.
hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.

Public Methods

BroadcastMessage	Calls the method named methodName on every MonoBehaviour in this game object or any of its children.
CompareTag	Is this game object tagged with tag ?
GetComponent	Returns the component of Type type if the game object has one attached, null if it doesn't.
GetComponentInChildren	Returns the component of Type type in the GameObject or any of its children using depth first search.
GetComponentInParent	Returns the component of Type type in the GameObject or any of its parents.
GetComponents	Returns all components of Type type in the GameObject.
GetComponentsInChildren	Returns all components of Type type in the GameObject or any of its children.
GetComponentsInParent	Returns all components of Type type in the GameObject or any of its parents.
SendMessage	Calls the method named methodName on every MonoBehaviour in this game

	object.
SendMessageUpwards	Calls the method named methodName on every MonoBehaviour in this game object and on every ancestor of the behaviour.
CancelInvoke	Cancels all Invoke calls on this MonoBehaviour.
Invoke	Invokes the method methodName in time seconds.
InvokeRepeating	Invokes the method methodName in time seconds, then repeatedly every repeatRate seconds.
IsInvoking	Is any invoke on methodName pending?
StartCoroutine	Starts a coroutine.
StopAllCoroutines	Stops all coroutines running on this behaviour.
StopCoroutine	Stops the first coroutine named methodName, or the coroutine stored in routine running on this behaviour.
GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.
IsActive	Returns true if the GameObject and the Component are active.
IsDestroyed	Returns true if the native representation of the behaviour has been destroyed.

Protected Methods

Awake	See MonoBehaviour.Awake.
OnBeforeTransformParentChanged	See MonoBehaviour.OnBeforeTransformParentChanged.
OnCanvasGroupChanged	See MonoBehaviour.OnCanvasGroupChanged.
OnCanvasHierarchyChanged	Called when the state of the parent Canvas is changed.
OnDestroy	See MonoBehaviour.OnDestroy.
OnDidApplyAnimationProperties	See LayoutGroup.OnDidApplyAnimationProperties.
OnEnable	See MonoBehaviour.OnEnable.
OnRectTransformDimensionsChange	This callback is called if an associated RectTransform has its dimensions changed.
OnTransformParentChanged	See MonoBehaviour.OnRectTransformParentChanged.
OnValidate	See MonoBehaviour.OnValidate.
Reset	See MonoBehaviour.Reset.

Start	See MonoBehaviour.Start.

Static Methods

print	Logs message to the Unity Console (identical to Debug.Log).
Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Messages

Awake	Awake is called when the script instance is being loaded.
FixedUpdate	This function is called every fixed framerate frame, if the MonoBehaviour is enabled.
LateUpdate	LateUpdate is called every frame, if the Behaviour is enabled.
OnAnimatorIK	Callback for setting up animation IK (inverse kinematics).
OnAnimatorMove	Callback for processing animation movements for modifying root motion.
OnApplicationFocus	Sent to all GameObjects when the player gets or loses focus.
OnApplicationPause	Sent to all GameObjects when the application pauses.
OnApplicationQuit	Sent to all game objects before the application quits.
OnAudioFilterRead	If OnAudioFilterRead is implemented, Unity will insert a custom filter into the audio DSP chain.

OnBecameInvisible	OnBecameInvisible is called when the renderer is no longer visible by any camera.
OnBecameVisible	OnBecameVisible is called when the renderer became visible by any camera.
OnCollisionEnter	OnCollisionEnter is called when this collider/rigidbody has begun touching another rigidbody/collider.
OnCollisionEnter2D	Sent when an incoming collider makes contact with this object's collider (2D physics only).
OnCollisionExit	OnCollisionExit is called when this collider/rigidbody has stopped touching another rigidbody/collider.
OnCollisionExit2D	Sent when a collider on another object stops touching this object's collider (2D physics only).
OnCollisionStay	:ref::OnCollisionStay is called once per frame for every collider/rigidbody that is touching rigidbody/collider.
OnCollisionStay2D	Sent each frame where a collider on another object is touching this object's collider (2D physics only).
OnConnectedToServer	Called on the client when you have successfully connected to a server.
OnControllerColliderHit	OnControllerColliderHit is called when the controller hits a collider while performing a Move.
OnDestroy	Destroying the attached Behaviour will result in the game or Scene receiving OnDestroy.
OnDisconnectedFromServer	Called on the client when the connection was lost or you disconnected from the server.
OnDrawGizmos	Implement OnDrawGizmos if you want to draw gizmos that are also pickable and always drawn.
OnDrawGizmosSelected	Implement OnDrawGizmosSelected to draw a gizmo if the object is selected.
OnEnable	This function is called when the object becomes enabled and active.
OnFailedToConnect	Called on the client when a connection attempt fails for some reason.
OnFailedToConnectToMasterServer	Called on clients or servers when there is a problem connecting to the MasterServer.
OnGUI	OnGUI is called for rendering and handling GUI events.
OnJointBreak	Called when a joint attached to the same game object broke.
OnJointBreak2D	Called when a Joint2D attached to the same game object breaks.

OnMasterServerEvent	Called on clients or servers when reporting events from the MasterServer.
OnMouseDown	OnMouseDown is called when the user has pressed the mouse button while over the GUIElement or Collider.
OnMouseDrag	OnMouseDrag is called when the user has clicked on a GUIElement or Collider and is still holding down the mouse.
OnMouseEnter	Called when the mouse enters the GUIElement or Collider.
OnMouseExit	Called when the mouse is not any longer over the GUIElement or Collider.
OnMouseOver	Called every frame while the mouse is over the GUIElement or Collider.
OnMouseUp	OnMouseUp is called when the user has released the mouse button.
OnMouseUpAsButton	OnMouseUpAsButton is only called when the mouse is released over the same GUIElement or Collider as it was pressed.
OnNetworkInstantiate	Called on objects which have been network instantiated with Network.Instantiate.
OnParticleCollision	OnParticleCollision is called when a particle hits a Collider.
OnParticleSystemStopped	OnParticleSystemStopped is called when all particles in the system have died, and no new particles will be born. New particles cease to be created either after Stop is called, or when the duration property of a non-looping system has been exceeded.
OnParticleTrigger	OnParticleTrigger is called when any particles in a particle system meet the conditions in the trigger module.
OnPlayerConnected	Called on the server whenever a new player has successfully connected.
OnPlayerDisconnected	Called on the server whenever a player disconnected from the server.
OnPostRender	OnPostRender is called after a camera finished rendering the Scene.
OnPreCull	OnPreCull is called before a camera culls the Scene.
OnPreRender	OnPreRender is called before a camera starts rendering the Scene.
OnRenderImage	OnRenderImage is called after all rendering is complete to render image.
OnRenderObject	OnRenderObject is called after camera has rendered the Scene.
OnSerializeNetworkView	Used to customize synchronization of variables in a script watched by a network view.

OnServerInitialized	Called on the server whenever a Network.InitializeServer was invoked and has completed.
OnTransformChildrenChanged	This function is called when the list of children of the transform of the GameObject has changed.
OnTransformParentChanged	This function is called when the parent property of the transform of the GameObject has changed.
OnTriggerEnter	OnTriggerEnter is called when the Collider other enters the trigger.
OnTriggerEnter2D	Sent when another object enters a trigger collider attached to this object (2D physics only).
OnTriggerExit	OnTriggerExit is called when the Collider other has stopped touching the trigger.
OnTriggerExit2D	Sent when another object leaves a trigger collider attached to this object (2D physics only).
OnTriggerStay	OnTriggerStay is called once per physics update for every Collider other that is touching the trigger.
OnTriggerStay2D	Sent each frame where another object is within a trigger collider attached to this object (2D physics only).
OnValidate	This function is called when the script is loaded or a value is changed in the inspector (Called in the editor only).
OnWillRenderObject	OnWillRenderObject is called for each camera if the object is visible and not a UI element.
Reset	Reset to default values.
Start	Start is called on the frame when a script is enabled just before any of the Update methods are called the first time.
Update	Update is called every frame, if the MonoBehaviour is enabled.

Implements interfaces:IBeginDragHandler, ICancelHandler, IDeselectHandler, IDragHandler, IDropHandler, IEndDragHandler, IEventSystemHandler, IInitializePotentialDragHandler, IMoveHandler, IPointerClickHandler, IPointerDownHandler, IPointerEnterHandler, IPointerExitHandler, IPointerUpHandler, IScrollHandler, ISelectHandler, ISubmitHandler, IUpdateSelectedHandler

Description

Receives events from the EventSystem and calls registered functions for each event.

The EventTrigger can be used to specify functions you wish to be called for each EventSystem event. You can assign multiple functions to a single event and whenever the EventTrigger receives that event it will call those functions in the order they were provided.

NOTE: Attaching this component to a GameObject will make that object intercept ALL events, and no events will propagate to parent objects.

There are two ways to intercept events: You could extend EventTrigger, and override the functions for the events you are interested in intercepting; as shown in this example:

```
using UnityEngine;
using UnityEngine.EventSystems;


public class EventTriggerExample : EventTrigger
{
    public override void OnBeginDrag(PointerEventData data)
    {
        Debug.Log("OnBeginDrag called.");
    }


    public override void OnCancel(BaseEventData data)
    {
        Debug.Log("OnCancel called.");
    }


    public override void OnDeselect(BaseEventData data)
    {
        Debug.Log("OnDeselect called.");
    }


    public override void OnDrag(PointerEventData data)
```

```
    {
        Debug.Log("OnDrag called.");
    }


    public override void OnDrop(PointerEventData data)
    {
        Debug.Log("OnDrop called.");
    }


    public override void OnEndDrag(PointerEventData data)
    {
        Debug.Log("OnEndDrag called.");
    }


    public override void OnInitializePotentialDrag(PointerEventData data)
    {
        Debug.Log("OnInitializePotentialDrag called.");
    }


    public override void OnMove(AxisEventData data)
    {
        Debug.Log("OnMove called.");
    }


    public override void OnPointerClick(PointerEventData data)
    {
        Debug.Log("OnPointerClick called.");
    }


    public override void OnPointerDown(PointerEventData data)
    {
        Debug.Log("OnPointerDown called.");
    }


    public override void OnPointerEnter(PointerEventData data)
    {
        Debug.Log("OnPointerEnter called.");
    }
```

```
    public override void OnPointerExit(PointerEventData data)
    {
        Debug.Log("OnPointerExit called.");
    }


    public override void OnPointerUp(PointerEventData data)
    {
        Debug.Log("OnPointerUp called.");
    }


    public override void OnScroll(PointerEventData data)
    {
        Debug.Log("OnScroll called.");
    }


    public override void OnSelect(BaseEventData data)
    {
        Debug.Log("OnSelect called.");
    }


    public override void OnSubmit(BaseEventData data)
    {
        Debug.Log("OnSubmit called.");
    }


    public override void OnUpdateSelected(BaseEventData data)
    {
        Debug.Log("OnUpdateSelected called.");
    }
}
```

...or you can specify individual delegates:

```
using UnityEngine;
using UnityEngine.EventSystems;
```

```
public class EventTriggerDelegateExample : MonoBehaviour
{
    void Start()
    {
        EventTrigger trigger = GetComponent<EventTrigger>();
        EventTrigger.Entry entry = new EventTrigger.Entry();
        entry.eventID = EventTriggerType.PointerDown;
        entry.callback.AddListener((data) => { OnPointerDownDelegate((PointerEve
        trigger.triggers.Add(entry);
    }


    public void OnPointerDownDelegate(PointerEventData data)
    {
        Debug.Log("OnPointerDownDelegate called.");
    }
}
```

Properties

triggers	All the functions registered in this EventTrigger.

Public Methods

OnBeginDrag	Called before a drag is started.
OnCancel	Called by the EventSystem when a Cancel event occurs.
OnDeselect	Called by the EventSystem when a new object is being selected.
OnDrag	Called by the EventSystem every time the pointer is moved during dragging.
OnDrop	Called by the EventSystem when an object accepts a drop.
OnEndDrag	Called by the EventSystem once dragging ends.
OnInitializePotentialDrag	Called by the EventSystem when a drag has been found, but before it is valid to begin the drag.
OnMove	Called by the EventSystem when a Move event occurs.
OnPointerClick	Called by the EventSystem when a Click event occurs.
OnPointerDown	Called by the EventSystem when a PointerDown event occurs.
OnPointerEnter	Called by the EventSystem when the pointer enters the object associated with this EventTrigger.

OnPointerExit	Called by the EventSystem when the pointer exits the object associated with this EventTrigger.
OnPointerUp	Called by the EventSystem when a PointerUp event occurs.
OnScroll	Called by the EventSystem when a Scroll event occurs.
OnSelect	Called by the EventSystem when a Select event occurs.
OnSubmit	Called by the EventSystem when a Submit event occurs.
OnUpdateSelected	Called by the EventSystem when the object associated with this EventTrigger is updated.

Inherited Members

Properties

enabled	Enabled Behaviours are Updated, disabled Behaviours are not.
isActiveAndEnabled	Has the Behaviour had active and enabled called?
gameObject	The game object this component is attached to. A component is always attached to a game object.
tag	The tag of this game object.
transform	The Transform attached to this GameObject.
runInEditMode	Allow a specific instance of a MonoBehaviour to run in edit mode (only available in the editor).
useGUILayout	Disabling this lets you skip the GUI layout phase.
hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.

Public Methods

BroadcastMessage	Calls the method named methodName on every MonoBehaviour in this game object or any of its children.
CompareTag	Is this game object tagged with tag ?
GetComponent	Returns the component of Type type if the game object has one attached, null if it doesn't.
GetComponentInChildren	Returns the component of Type type in the GameObject or any of its children using depth first search.

GetComponentInParent	Returns the component of Type type in the GameObject or any of its parents.
GetComponents	Returns all components of Type type in the GameObject.
GetComponentsInChildren	Returns all components of Type type in the GameObject or any of its children.
GetComponentsInParent	Returns all components of Type type in the GameObject or any of its parents.
SendMessage	Calls the method named methodName on every MonoBehaviour in this game object.
SendMessageUpwards	Calls the method named methodName on every MonoBehaviour in this game object and on every ancestor of the behaviour.
CancelInvoke	Cancels all Invoke calls on this MonoBehaviour.
Invoke	Invokes the method methodName in time seconds.
InvokeRepeating	Invokes the method methodName in time seconds, then repeatedly every repeatRate seconds.
IsInvoking	Is any invoke on methodName pending?
StartCoroutine	Starts a coroutine.
StopAllCoroutines	Stops all coroutines running on this behaviour.
StopCoroutine	Stops the first coroutine named methodName, or the coroutine stored in routine running on this behaviour.
GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.

Static Methods

print	Logs message to the Unity Console (identical to Debug.Log).
Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Messages

Awake	Awake is called when the script instance is being loaded.
FixedUpdate	This function is called every fixed framerate frame, if the MonoBehaviour is enabled.
LateUpdate	LateUpdate is called every frame, if the Behaviour is enabled.
OnAnimatorIK	Callback for setting up animation IK (inverse kinematics).
OnAnimatorMove	Callback for processing animation movements for modifying root motion.
OnApplicationFocus	Sent to all GameObjects when the player gets or loses focus.
OnApplicationPause	Sent to all GameObjects when the application pauses.
OnApplicationQuit	Sent to all game objects before the application quits.
OnAudioFilterRead	If OnAudioFilterRead is implemented, Unity will insert a custom filter into the audio DSP chain.
OnBecameInvisible	OnBecameInvisible is called when the renderer is no longer visible by any camera.
OnBecameVisible	OnBecameVisible is called when the renderer became visible by any camera.
OnCollisionEnter	OnCollisionEnter is called when this collider/rigidbody has begun touching another rigidbody/collider.
OnCollisionEnter2D	Sent when an incoming collider makes contact with this object's collider (2D physics only).
OnCollisionExit	OnCollisionExit is called when this collider/rigidbody has stopped touching another rigidbody/collider.
OnCollisionExit2D	Sent when a collider on another object stops touching this object's collider (2D physics only).
OnCollisionStay	:ref::OnCollisionStay is called once per frame for every collider/rigidbody that is touching rigidbody/collider.
OnCollisionStay2D	Sent each frame where a collider on another object is touching this

	object's collider (2D physics only).
OnConnectedToServer	Called on the client when you have successfully connected to a server.
OnControllerColliderHit	OnControllerColliderHit is called when the controller hits a collider while performing a Move.
OnDestroy	Destroying the attached Behaviour will result in the game or Scene receiving OnDestroy.
OnDisable	This function is called when the behaviour becomes disabled.
OnDisconnectedFromServer	Called on the client when the connection was lost or you disconnected from the server.
OnDrawGizmos	Implement OnDrawGizmos if you want to draw gizmos that are also pickable and always drawn.
OnDrawGizmosSelected	Implement OnDrawGizmosSelected to draw a gizmo if the object is selected.
OnEnable	This function is called when the object becomes enabled and active.
OnFailedToConnect	Called on the client when a connection attempt fails for some reason.
OnFailedToConnectToMasterServer	Called on clients or servers when there is a problem connecting to the MasterServer.
OnGUI	OnGUI is called for rendering and handling GUI events.
OnJointBreak	Called when a joint attached to the same game object broke.
OnJointBreak2D	Called when a Joint2D attached to the same game object breaks.
OnMasterServerEvent	Called on clients or servers when reporting events from the MasterServer.
OnMouseDown	OnMouseDown is called when the user has pressed the mouse button while over the GUIElement or Collider.
OnMouseDrag	OnMouseDrag is called when the user has clicked on a GUIElement or Collider and is still holding down the mouse.
OnMouseEnter	Called when the mouse enters the GUIElement or Collider.
OnMouseExit	Called when the mouse is not any longer over the GUIElement or Collider.
OnMouseOver	Called every frame while the mouse is over the GUIElement or Collider.
OnMouseUp	OnMouseUp is called when the user has released the mouse button.
OnMouseUpAsButton	OnMouseUpAsButton is only called when the mouse is released over

	the same GUIElement or Collider as it was pressed.
OnNetworkInstantiate	Called on objects which have been network instantiated with Network.Instantiate.
OnParticleCollision	OnParticleCollision is called when a particle hits a Collider.
OnParticleSystemStopped	OnParticleSystemStopped is called when all particles in the system have died, and no new particles will be born. New particles cease to be created either after Stop is called, or when the duration property of a non-looping system has been exceeded.
OnParticleTrigger	OnParticleTrigger is called when any particles in a particle system meet the conditions in the trigger module.
OnPlayerConnected	Called on the server whenever a new player has successfully connected.
OnPlayerDisconnected	Called on the server whenever a player disconnected from the server.
OnPostRender	OnPostRender is called after a camera finished rendering the Scene.
OnPreCull	OnPreCull is called before a camera culls the Scene.
OnPreRender	OnPreRender is called before a camera starts rendering the Scene.
OnRenderImage	OnRenderImage is called after all rendering is complete to render image.
OnRenderObject	OnRenderObject is called after camera has rendered the Scene.
OnSerializeNetworkView	Used to customize synchronization of variables in a script watched by a network view.
OnServerInitialized	Called on the server whenever a Network.InitializeServer was invoked and has completed.
OnTransformChildrenChanged	This function is called when the list of children of the transform of the GameObject has changed.
OnTransformParentChanged	This function is called when the parent property of the transform of the GameObject has changed.
OnTriggerEnter	OnTriggerEnter is called when the Collider other enters the trigger.
OnTriggerEnter2D	Sent when another object enters a trigger collider attached to this object (2D physics only).
OnTriggerExit	OnTriggerExit is called when the Collider other has stopped touching the trigger.
OnTriggerExit2D	Sent when another object leaves a trigger collider attached to this object (2D physics only).

OnTriggerStay	OnTriggerStay is called once per physics update for every Collider other that is touching the trigger.
OnTriggerStay2D	Sent each frame where another object is within a trigger collider attached to this object (2D physics only).
OnValidate	This function is called when the script is loaded or a value is changed in the inspector (Called in the editor only).
OnWillRenderObject	OnWillRenderObject is called for each camera if the object is visible and not a UI element.
Reset	Reset to default values.
Start	Start is called on the frame when a script is enabled just before any of the Update methods are called the first time.
Update	Update is called every frame, if the MonoBehaviour is enabled.

Description

An Entry in the EventSystem delegates list.

It stores the callback and which event type should this callback be fired.

Properties

callback	The desired TriggerEvent to be Invoked.
eventID	What type of event is the associated callback listening for.

Description

UnityEvent class for Triggers.

See EventTrigger for example usage.

Inherited Members

Public Methods

GetPersistentEventCount	Get the number of registered persistent listeners.
GetPersistentMethodName	Get the target method name of the listener at index index.
GetPersistentTarget	Get the target component of the listener at index index.
RemoveAllListeners	Remove all non-persisent (ie created from script) listeners from the event.
SetPersistentListenerState	Modify the execution state of a persistent listener.

Static Methods

GetValidMethodInfo	Given an object, function name, and a list of argument types; find the method that matches.

Description

Helper class that can be used to send IEventSystemHandler events to GameObjects.

Static Properties

beginDragHandler	IBeginDragHandler execute helper function.
cancelHandler	ICancelHandler execute helper function.
deselectHandler	IDeselectHandler execute helper function.
dragHandler	IDragHandler execute helper function.
dropHandler	IDropHandler execute helper function.
endDragHandler	IEndDragHandler execute helper function.
initializePotentialDrag	IInitializePotentialDragHandler execute helper function.
moveHandler	IMoveHandler execute helper function.
pointerClickHandler	IPointerClickHandler execute helper function.
pointerDownHandler	IPointerDownHandler execute helper function.
pointerEnterHandler	IPointerEnterHandler execute helper function.
pointerExitHandler	IPointerExitHandler execute helper function.
pointerUpHandler	IPointerUpHandler execute helper function.
scrollHandler	IScrollHandler execute helper function.
selectHandler	ISelectHandler execute helper function.
submitHandler	ISubmitHandler execute helper function.
updateSelectedHandler	IUpdateSelectedHandler execute helper function.

Static Methods

CanHandleEvent	Can the given GameObject handle the IEventSystemHandler of type T.
Execute	Execute the event of type T : IEventSystemHandler on GameObject.
ExecuteHierarchy	Recurse up the hierarchy calling Execute<T> until there is a GameObject that can handle the event.

GetEventHandler	Traverse the object hierarchy starting at root, and return the GameObject which implements the event handler of type <T>.
ValidateEventData	Attempt to convert the data to type T. If conversion fails an ArgumentException is thrown.

Delegates

EventFunction<T0>	Function definition for an EventFunction that is used to map a given EventInterface into a specific event call.

Description

Raycaster for casting against 2D Physics components.

See: BaseRaycaster.

Public Methods

Raycast	See: BaseRaycaster.

Inherited Members

Properties

eventCamera	The camera that will generate rays for this raycaster.
renderOrderPriority	Priority of the raycaster based upon render order.
sortOrderPriority	Priority of the raycaster based upon sort order.
enabled	Enabled Behaviours are Updated, disabled Behaviours are not.
isActiveAndEnabled	Has the Behaviour had active and enabled called?
gameObject	The game object this component is attached to. A component is always attached to a game object.
tag	The tag of this game object.
transform	The Transform attached to this GameObject.
runInEditMode	Allow a specific instance of a MonoBehaviour to run in edit mode (only available in the editor).
useGUILayout	Disabling this lets you skip the GUI layout phase.
hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.
depth	Get the depth of the configured camera.
eventCamera	Get the camera that is used for this module.
eventMask	Mask of allowed raycast events.
finalEventMask	Logical and of Camera mask and eventMask.
maxRayIntersections	Max number of ray intersection allowed to be found.

Public Methods

BroadcastMessage	Calls the method named methodName on every MonoBehaviour in this game object or any of its children.
CompareTag	Is this game object tagged with tag ?
GetComponent	Returns the component of Type type if the game object has one attached, null if it doesn't.
GetComponentInChildren	Returns the component of Type type in the GameObject or any of its children using depth first search.
GetComponentInParent	Returns the component of Type type in the GameObject or any of its parents.
GetComponents	Returns all components of Type type in the GameObject.
GetComponentsInChildren	Returns all components of Type type in the GameObject or any of its children.
GetComponentsInParent	Returns all components of Type type in the GameObject or any of its parents.
SendMessage	Calls the method named methodName on every MonoBehaviour in this game object.
SendMessageUpwards	Calls the method named methodName on every MonoBehaviour in this game object and on every ancestor of the behaviour.
CancelInvoke	Cancels all Invoke calls on this MonoBehaviour.
Invoke	Invokes the method methodName in time seconds.
InvokeRepeating	Invokes the method methodName in time seconds, then repeatedly every repeatRate seconds.
IsInvoking	Is any invoke on methodName pending?
StartCoroutine	Starts a coroutine.
StopAllCoroutines	Stops all coroutines running on this behaviour.
StopCoroutine	Stops the first coroutine named methodName, or the coroutine stored in routine running on this behaviour.
GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.
IsActive	Returns true if the GameObject and the Component are active.
IsDestroyed	Returns true if the native representation of the behaviour has been destroyed.

Protected Methods

OnDisable	See MonoBehaviour.OnDisable.
ComputeRayAndDistance	Returns a ray going from camera through the event position and the distance between the near and far clipping planes along that ray.
Awake	See MonoBehaviour.Awake.
OnBeforeTransformParentChanged	See MonoBehaviour.OnBeforeTransformParentChanged.
OnCanvasGroupChanged	See MonoBehaviour.OnCanvasGroupChanged.
OnCanvasHierarchyChanged	Called when the state of the parent Canvas is changed.
OnDestroy	See MonoBehaviour.OnDestroy.
OnDidApplyAnimationProperties	See LayoutGroup.OnDidApplyAnimationProperties.
OnDisable	See MonoBehaviour.OnDisable.
OnEnable	See MonoBehaviour.OnEnable.
OnRectTransformDimensionsChange	This callback is called if an associated RectTransform has its dimensions changed.
OnTransformParentChanged	See MonoBehaviour.OnRectTransformParentChanged.
OnValidate	See MonoBehaviour.OnValidate.
Reset	See MonoBehaviour.Reset.
Start	See MonoBehaviour.Start.

Static Methods

print	Logs message to the Unity Console (identical to Debug.Log).
Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Messages

Awake	Awake is called when the script instance is being loaded.
FixedUpdate	This function is called every fixed framerate frame, if the MonoBehaviour is enabled.
LateUpdate	LateUpdate is called every frame, if the Behaviour is enabled.
OnAnimatorIK	Callback for setting up animation IK (inverse kinematics).
OnAnimatorMove	Callback for processing animation movements for modifying root motion.
OnApplicationFocus	Sent to all GameObjects when the player gets or loses focus.
OnApplicationPause	Sent to all GameObjects when the application pauses.
OnApplicationQuit	Sent to all game objects before the application quits.
OnAudioFilterRead	If OnAudioFilterRead is implemented, Unity will insert a custom filter into the audio DSP chain.
OnBecameInvisible	OnBecameInvisible is called when the renderer is no longer visible by any camera.
OnBecameVisible	OnBecameVisible is called when the renderer became visible by any camera.
OnCollisionEnter	OnCollisionEnter is called when this collider/rigidbody has begun touching another rigidbody/collider.
OnCollisionEnter2D	Sent when an incoming collider makes contact with this object's collider (2D physics only).
OnCollisionExit	OnCollisionExit is called when this collider/rigidbody has stopped touching another rigidbody/collider.
OnCollisionExit2D	Sent when a collider on another object stops touching this object's collider (2D physics only).
OnCollisionStay	:ref::OnCollisionStay is called once per frame for every collider/rigidbody that is touching rigidbody/collider.
OnCollisionStay2D	Sent each frame where a collider on another object is touching this

	object's collider (2D physics only).
OnConnectedToServer	Called on the client when you have successfully connected to a server.
OnControllerColliderHit	OnControllerColliderHit is called when the controller hits a collider while performing a Move.
OnDestroy	Destroying the attached Behaviour will result in the game or Scene receiving OnDestroy.
OnDisable	This function is called when the behaviour becomes disabled.
OnDisconnectedFromServer	Called on the client when the connection was lost or you disconnected from the server.
OnDrawGizmos	Implement OnDrawGizmos if you want to draw gizmos that are also pickable and always drawn.
OnDrawGizmosSelected	Implement OnDrawGizmosSelected to draw a gizmo if the object is selected.
OnEnable	This function is called when the object becomes enabled and active.
OnFailedToConnect	Called on the client when a connection attempt fails for some reason.
OnFailedToConnectToMasterServer	Called on clients or servers when there is a problem connecting to the MasterServer.
OnGUI	OnGUI is called for rendering and handling GUI events.
OnJointBreak	Called when a joint attached to the same game object broke.
OnJointBreak2D	Called when a Joint2D attached to the same game object breaks.
OnMasterServerEvent	Called on clients or servers when reporting events from the MasterServer.
OnMouseDown	OnMouseDown is called when the user has pressed the mouse button while over the GUIElement or Collider.
OnMouseDrag	OnMouseDrag is called when the user has clicked on a GUIElement or Collider and is still holding down the mouse.
OnMouseEnter	Called when the mouse enters the GUIElement or Collider.
OnMouseExit	Called when the mouse is not any longer over the GUIElement or Collider.
OnMouseOver	Called every frame while the mouse is over the GUIElement or Collider.
OnMouseUp	OnMouseUp is called when the user has released the mouse button.
OnMouseUpAsButton	OnMouseUpAsButton is only called when the mouse is released over

	the same GUIElement or Collider as it was pressed.
OnNetworkInstantiate	Called on objects which have been network instantiated with Network.Instantiate.
OnParticleCollision	OnParticleCollision is called when a particle hits a Collider.
OnParticleSystemStopped	OnParticleSystemStopped is called when all particles in the system have died, and no new particles will be born. New particles cease to be created either after Stop is called, or when the duration property of a non-looping system has been exceeded.
OnParticleTrigger	OnParticleTrigger is called when any particles in a particle system meet the conditions in the trigger module.
OnPlayerConnected	Called on the server whenever a new player has successfully connected.
OnPlayerDisconnected	Called on the server whenever a player disconnected from the server.
OnPostRender	OnPostRender is called after a camera finished rendering the Scene.
OnPreCull	OnPreCull is called before a camera culls the Scene.
OnPreRender	OnPreRender is called before a camera starts rendering the Scene.
OnRenderImage	OnRenderImage is called after all rendering is complete to render image.
OnRenderObject	OnRenderObject is called after camera has rendered the Scene.
OnSerializeNetworkView	Used to customize synchronization of variables in a script watched by a network view.
OnServerInitialized	Called on the server whenever a Network.InitializeServer was invoked and has completed.
OnTransformChildrenChanged	This function is called when the list of children of the transform of the GameObject has changed.
OnTransformParentChanged	This function is called when the parent property of the transform of the GameObject has changed.
OnTriggerEnter	OnTriggerEnter is called when the Collider other enters the trigger.
OnTriggerEnter2D	Sent when another object enters a trigger collider attached to this object (2D physics only).
OnTriggerExit	OnTriggerExit is called when the Collider other has stopped touching the trigger.
OnTriggerExit2D	Sent when another object leaves a trigger collider attached to this object (2D physics only).

OnTriggerStay	OnTriggerStay is called once per physics update for every Collider other that is touching the trigger.
OnTriggerStay2D	Sent each frame where another object is within a trigger collider attached to this object (2D physics only).
OnValidate	This function is called when the script is loaded or a value is changed in the inspector (Called in the editor only).
OnWillRenderObject	OnWillRenderObject is called for each camera if the object is visible and not a UI element.
Reset	Reset to default values.
Start	Start is called on the frame when a script is enabled just before any of the Update methods are called the first time.
Update	Update is called every frame, if the MonoBehaviour is enabled.

Description

Raycaster for casting against 3D Physics components.

See: BaseRaycaster.

Properties

depth	Get the depth of the configured camera.
eventCamera	Get the camera that is used for this module.
eventMask	Mask of allowed raycast events.
finalEventMask	Logical and of Camera mask and eventMask.
maxRayIntersections	Max number of ray intersection allowed to be found.

Public Methods

Raycast	See: BaseRaycaster.

Protected Methods

ComputeRayAndDistance	Returns a ray going from camera through the event position and the distance between the near and far clipping planes along that ray.

Inherited Members

Properties

eventCamera	The camera that will generate rays for this raycaster.
renderOrderPriority	Priority of the raycaster based upon render order.
sortOrderPriority	Priority of the raycaster based upon sort order.
enabled	Enabled Behaviours are Updated, disabled Behaviours are not.
isActiveAndEnabled	Has the Behaviour had active and enabled called?
gameObject	The game object this component is attached to. A component is always attached to a game object.
tag	The tag of this game object.

transform	The Transform attached to this GameObject.
runInEditMode	Allow a specific instance of a MonoBehaviour to run in edit mode (only available in the editor).
useGUILayout	Disabling this lets you skip the GUI layout phase.
hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.

Public Methods

BroadcastMessage	Calls the method named methodName on every MonoBehaviour in this game object or any of its children.
CompareTag	Is this game object tagged with tag ?
GetComponent	Returns the component of Type type if the game object has one attached, null if it doesn't.
GetComponentInChildren	Returns the component of Type type in the GameObject or any of its children using depth first search.
GetComponentInParent	Returns the component of Type type in the GameObject or any of its parents.
GetComponents	Returns all components of Type type in the GameObject.
GetComponentsInChildren	Returns all components of Type type in the GameObject or any of its children.
GetComponentsInParent	Returns all components of Type type in the GameObject or any of its parents.
SendMessage	Calls the method named methodName on every MonoBehaviour in this game object.
SendMessageUpwards	Calls the method named methodName on every MonoBehaviour in this game object and on every ancestor of the behaviour.
CancelInvoke	Cancels all Invoke calls on this MonoBehaviour.
Invoke	Invokes the method methodName in time seconds.
InvokeRepeating	Invokes the method methodName in time seconds, then repeatedly every repeatRate seconds.
IsInvoking	Is any invoke on methodName pending?
StartCoroutine	Starts a coroutine.
StopAllCoroutines	Stops all coroutines running on this behaviour.
StopCoroutine	Stops the first coroutine named methodName, or the coroutine stored in routine

	running on this behaviour.
GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.
IsActive	Returns true if the GameObject and the Component are active.
IsDestroyed	Returns true if the native representation of the behaviour has been destroyed.

Protected Methods

OnDisable	See MonoBehaviour.OnDisable.
Awake	See MonoBehaviour.Awake.
OnBeforeTransformParentChanged	See MonoBehaviour.OnBeforeTransformParentChanged.
OnCanvasGroupChanged	See MonoBehaviour.OnCanvasGroupChanged.
OnCanvasHierarchyChanged	Called when the state of the parent Canvas is changed.
OnDestroy	See MonoBehaviour.OnDestroy.
OnDidApplyAnimationProperties	See LayoutGroup.OnDidApplyAnimationProperties.
OnDisable	See MonoBehaviour.OnDisable.
OnEnable	See MonoBehaviour.OnEnable.
OnRectTransformDimensionsChange	This callback is called if an associated RectTransform has its dimensions changed.
OnTransformParentChanged	See MonoBehaviour.OnRectTransformParentChanged.
OnValidate	See MonoBehaviour.OnValidate.
Reset	See MonoBehaviour.Reset.
Start	See MonoBehaviour.Start.

Static Methods

print	Logs message to the Unity Console (identical to Debug.Log).
Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.

FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Messages

Awake	Awake is called when the script instance is being loaded.
FixedUpdate	This function is called every fixed framerate frame, if the MonoBehaviour is enabled.
LateUpdate	LateUpdate is called every frame, if the Behaviour is enabled.
OnAnimatorIK	Callback for setting up animation IK (inverse kinematics).
OnAnimatorMove	Callback for processing animation movements for modifying root motion.
OnApplicationFocus	Sent to all GameObjects when the player gets or loses focus.
OnApplicationPause	Sent to all GameObjects when the application pauses.
OnApplicationQuit	Sent to all game objects before the application quits.
OnAudioFilterRead	If OnAudioFilterRead is implemented, Unity will insert a custom filter into the audio DSP chain.
OnBecameInvisible	OnBecameInvisible is called when the renderer is no longer visible by any camera.
OnBecameVisible	OnBecameVisible is called when the renderer became visible by any camera.
OnCollisionEnter	OnCollisionEnter is called when this collider/rigidbody has begun touching another rigidbody/collider.
OnCollisionEnter2D	Sent when an incoming collider makes contact with this object's collider (2D physics only).
OnCollisionExit	OnCollisionExit is called when this collider/rigidbody has stopped touching another rigidbody/collider.

OnCollisionExit2D	Sent when a collider on another object stops touching this object's collider (2D physics only).
OnCollisionStay	:ref::OnCollisionStay is called once per frame for every collider/rigidbody that is touching rigidbody/collider.
OnCollisionStay2D	Sent each frame where a collider on another object is touching this object's collider (2D physics only).
OnConnectedToServer	Called on the client when you have successfully connected to a server.
OnControllerColliderHit	OnControllerColliderHit is called when the controller hits a collider while performing a Move.
OnDestroy	Destroying the attached Behaviour will result in the game or Scene receiving OnDestroy.
OnDisable	This function is called when the behaviour becomes disabled.
OnDisconnectedFromServer	Called on the client when the connection was lost or you disconnected from the server.
OnDrawGizmos	Implement OnDrawGizmos if you want to draw gizmos that are also pickable and always drawn.
OnDrawGizmosSelected	Implement OnDrawGizmosSelected to draw a gizmo if the object is selected.
OnEnable	This function is called when the object becomes enabled and active.
OnFailedToConnect	Called on the client when a connection attempt fails for some reason.
OnFailedToConnectToMasterServer	Called on clients or servers when there is a problem connecting to the MasterServer.
OnGUI	OnGUI is called for rendering and handling GUI events.
OnJointBreak	Called when a joint attached to the same game object broke.
OnJointBreak2D	Called when a Joint2D attached to the same game object breaks.
OnMasterServerEvent	Called on clients or servers when reporting events from the MasterServer.
OnMouseDown	OnMouseDown is called when the user has pressed the mouse button while over the GUIElement or Collider.
OnMouseDrag	OnMouseDrag is called when the user has clicked on a GUIElement or Collider and is still holding down the mouse.
OnMouseEnter	Called when the mouse enters the GUIElement or Collider.

OnMouseExit	Called when the mouse is not any longer over the GUIElement or Collider.
OnMouseOver	Called every frame while the mouse is over the GUIElement or Collider.
OnMouseUp	OnMouseUp is called when the user has released the mouse button.
OnMouseUpAsButton	OnMouseUpAsButton is only called when the mouse is released over the same GUIElement or Collider as it was pressed.
OnNetworkInstantiate	Called on objects which have been network instantiated with Network.Instantiate.
OnParticleCollision	OnParticleCollision is called when a particle hits a Collider.
OnParticleSystemStopped	OnParticleSystemStopped is called when all particles in the system have died, and no new particles will be born. New particles cease to be created either after Stop is called, or when the duration property of a non-looping system has been exceeded.
OnParticleTrigger	OnParticleTrigger is called when any particles in a particle system meet the conditions in the trigger module.
OnPlayerConnected	Called on the server whenever a new player has successfully connected.
OnPlayerDisconnected	Called on the server whenever a player disconnected from the server.
OnPostRender	OnPostRender is called after a camera finished rendering the Scene.
OnPreCull	OnPreCull is called before a camera culls the Scene.
OnPreRender	OnPreRender is called before a camera starts rendering the Scene.
OnRenderImage	OnRenderImage is called after all rendering is complete to render image.
OnRenderObject	OnRenderObject is called after camera has rendered the Scene.
OnSerializeNetworkView	Used to customize synchronization of variables in a script watched by a network view.
OnServerInitialized	Called on the server whenever a Network.InitializeServer was invoked and has completed.
OnTransformChildrenChanged	This function is called when the list of children of the transform of the GameObject has changed.
OnTransformParentChanged	This function is called when the parent property of the transform of the GameObject has changed.
OnTriggerEnter	OnTriggerEnter is called when the Collider other enters the trigger.
OnTriggerEnter2D	Sent when another object enters a trigger collider attached to this

	object (2D physics only).
OnTriggerExit	OnTriggerExit is called when the Collider other has stopped touching the trigger.
OnTriggerExit2D	Sent when another object leaves a trigger collider attached to this object (2D physics only).
OnTriggerStay	OnTriggerStay is called once per physics update for every Collider other that is touching the trigger.
OnTriggerStay2D	Sent each frame where another object is within a trigger collider attached to this object (2D physics only).
OnValidate	This function is called when the script is loaded or a value is changed in the inspector (Called in the editor only).
OnWillRenderObject	OnWillRenderObject is called for each camera if the object is visible and not a UI element.
Reset	Reset to default values.
Start	Start is called on the frame when a script is enabled just before any of the Update methods are called the first time.
Update	Update is called every frame, if the MonoBehaviour is enabled.

Description

Event payload associated with pointer (mouse / touch) events.

Properties

button	The InputButton for this event.
clickCount	Number of clicks in a row.
clickTime	The last time a click event was sent.
delta	Pointer delta since last update.
dragging	Is a drag operation currently occuring.
enterEventCamera	The camera associated with the last OnPointerEnter event.
hovered	List of objects in the hover stack.
lastPress	The GameObject for the last press event.
pointerCurrentRaycast	RaycastResult associated with the current event.
pointerDrag	The object that is receiving OnDrag.
pointerEnter	The object that received 'OnPointerEnter'.
pointerId	Identification of the pointer.
pointerPress	The GameObject that received the OnPointerDown.
pointerPressRaycast	RaycastResult associated with the pointer press.
position	Current pointer position.
pressEventCamera	The camera associated with the last OnPointerPress event.
pressPosition	The screen space coordinates of the last pointer click.
rawPointerPress	The object that the press happened on even if it can not handle the press event.
scrollDelta	The amount of scroll since the last update.
useDragThreshold	Should a drag threshold be used?

Public Methods

IsPointerMoving	Is the pointer moving.

IsScrolling	Is scroll being used on the input device.

Inherited Members

Properties

used	Is the event used?
currentInputModule	A reference to the BaseInputModule that sent this event.
selectedObject	The object currently considered selected by the EventSystem.

Public Methods

Reset	Reset the event.
Use	Use the event.

Description

The state of a press for the given frame.

Properties

Pressed	Button was pressed this frame.
Released	Button was released this frame.
PressedAndReleased	Button was pressed and released this frame.
NotChanged	Same as last frame.

Description

Input press tracking.

Properties

Left	Left button.
Right	Right button.
Middle	Middle button.

Description

A BaseInputModule for pointer input.

Used by TouchInputModule and StandaloneInputModule.

Static Properties

kFakeTouchesId	Touch id for when simulating touches on a non touch device.
kMouseLeftId	Id of the cached left mouse pointer event.
kMouseMiddleId	Id of the cached middle mouse pointer event.
kMouseRightId	Id of the cached right mouse pointer event.

Protected Methods

ClearSelection	Clear all pointers and deselect any selected objects in the EventSystem.
CopyFromTo	Copy one PointerEventData to another.
DeselectIfSelectionChanged	Deselect the current selected GameObject if the currently pointed-at GameObject is different.
GetLastPointerEventData	Return the last PointerEventData for the given touch / mouse id.
GetMousePointerEventData	Return the current MouseState.
GetPointerData	Search the cache for currently active pointers, return true if found.
GetTouchPointerEventData	Given a touch populate the PointerEventData and return if we are pressed or released.
ProcessDrag	Process the drag for the current frame with the given pointer event.
ProcessMove	Process movement for the current frame with the given pointer event.
RemovePointerData	Remove the PointerEventData from the cache.
StateForMouseButton	Given a mouse button return the current state for the frame.

Inherited Members

Properties

input	The current BaseInput being used by the input module.

inputOverride	Used to override the default BaseInput for the input module.
enabled	Enabled Behaviours are Updated, disabled Behaviours are not.
isActiveAndEnabled	Has the Behaviour had active and enabled called?
gameObject	The game object this component is attached to. A component is always attached to a game object.
tag	The tag of this game object.
transform	The Transform attached to this GameObject.
runInEditMode	Allow a specific instance of a MonoBehaviour to run in edit mode (only available in the editor).
useGUILayout	Disabling this lets you skip the GUI layout phase.
hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.

Public Methods

ActivateModule	Called when the module is activated. Override this if you want custom code to execute when you activate your module.
DeactivateModule	Called when the module is deactivated. Override this if you want custom code to execute when you deactivate your module.
IsModuleSupported	Check to see if the module is supported. Override this if you have a platfrom specific module (eg. TouchInputModule that you do not want to activate on standalone.
IsPointerOverGameObject	Is the pointer with the given ID over an EventSystem object?
Process	Process the current tick for the module.
ShouldActivateModule	Should be activated.
UpdateModule	Update the internal state of the Module.
BroadcastMessage	Calls the method named methodName on every MonoBehaviour in this game object or any of its children.
CompareTag	Is this game object tagged with tag ?
GetComponent	Returns the component of Type type if the game object has one attached, null if it doesn't.
GetComponentInChildren	Returns the component of Type type in the GameObject or any of its children

	using depth first search.
GetComponentInParent	Returns the component of Type type in the GameObject or any of its parents.
GetComponents	Returns all components of Type type in the GameObject.
GetComponentsInChildren	Returns all components of Type type in the GameObject or any of its children.
GetComponentsInParent	Returns all components of Type type in the GameObject or any of its parents.
SendMessage	Calls the method named methodName on every MonoBehaviour in this game object.
SendMessageUpwards	Calls the method named methodName on every MonoBehaviour in this game object and on every ancestor of the behaviour.
CancelInvoke	Cancels all Invoke calls on this MonoBehaviour.
Invoke	Invokes the method methodName in time seconds.
InvokeRepeating	Invokes the method methodName in time seconds, then repeatedly every repeatRate seconds.
IsInvoking	Is any invoke on methodName pending?
StartCoroutine	Starts a coroutine.
StopAllCoroutines	Stops all coroutines running on this behaviour.
StopCoroutine	Stops the first coroutine named methodName, or the coroutine stored in routine running on this behaviour.
GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.
IsActive	Returns true if the GameObject and the Component are active.
IsDestroyed	Returns true if the native representation of the behaviour has been destroyed.

Protected Methods

GetAxisEventData	Given some input data generate an AxisEventData that can be used by the event system.
GetBaseEventData	Generate a BaseEventData that can be used by the EventSystem.
HandlePointerExitAndEnter	Handle sending enter and exit events when a new enter target is found.
OnDisable	See MonoBehaviour.OnDisable.
OnEnable	See MonoBehaviour.OnEnable.

Awake	See MonoBehaviour.Awake.
OnBeforeTransformParentChanged	See MonoBehaviour.OnBeforeTransformParentChanged.
OnCanvasGroupChanged	See MonoBehaviour.OnCanvasGroupChanged.
OnCanvasHierarchyChanged	Called when the state of the parent Canvas is changed.
OnDestroy	See MonoBehaviour.OnDestroy.
OnDidApplyAnimationProperties	See LayoutGroup.OnDidApplyAnimationProperties.
OnDisable	See MonoBehaviour.OnDisable.
OnEnable	See MonoBehaviour.OnEnable.
OnRectTransformDimensionsChange	This callback is called if an associated RectTransform has its dimensions changed.
OnTransformParentChanged	See MonoBehaviour.OnRectTransformParentChanged.
OnValidate	See MonoBehaviour.OnValidate.
Reset	See MonoBehaviour.Reset.
Start	See MonoBehaviour.Start.

Static Methods

DetermineMoveDirection	Given an input movement, determine the best MoveDirection.
FindCommonRoot	Given 2 GameObjects, return a common root GameObject (or null).
FindFirstRaycast	Return the first valid RaycastResult.
print	Logs message to the Unity Console (identical to Debug.Log).
Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Messages

Awake	Awake is called when the script instance is being loaded.
FixedUpdate	This function is called every fixed framerate frame, if the MonoBehaviour is enabled.
LateUpdate	LateUpdate is called every frame, if the Behaviour is enabled.
OnAnimatorIK	Callback for setting up animation IK (inverse kinematics).
OnAnimatorMove	Callback for processing animation movements for modifying root motion.
OnApplicationFocus	Sent to all GameObjects when the player gets or loses focus.
OnApplicationPause	Sent to all GameObjects when the application pauses.
OnApplicationQuit	Sent to all game objects before the application quits.
OnAudioFilterRead	If OnAudioFilterRead is implemented, Unity will insert a custom filter into the audio DSP chain.
OnBecameInvisible	OnBecameInvisible is called when the renderer is no longer visible by any camera.
OnBecameVisible	OnBecameVisible is called when the renderer became visible by any camera.
OnCollisionEnter	OnCollisionEnter is called when this collider/rigidbody has begun touching another rigidbody/collider.
OnCollisionEnter2D	Sent when an incoming collider makes contact with this object's collider (2D physics only).
OnCollisionExit	OnCollisionExit is called when this collider/rigidbody has stopped touching another rigidbody/collider.
OnCollisionExit2D	Sent when a collider on another object stops touching this object's collider (2D physics only).
OnCollisionStay	:ref::OnCollisionStay is called once per frame for every collider/rigidbody that is touching rigidbody/collider.

OnCollisionStay2D	Sent each frame where a collider on another object is touching this object's collider (2D physics only).
OnConnectedToServer	Called on the client when you have successfully connected to a server.
OnControllerColliderHit	OnControllerColliderHit is called when the controller hits a collider while performing a Move.
OnDestroy	Destroying the attached Behaviour will result in the game or Scene receiving OnDestroy.
OnDisable	This function is called when the behaviour becomes disabled.
OnDisconnectedFromServer	Called on the client when the connection was lost or you disconnected from the server.
OnDrawGizmos	Implement OnDrawGizmos if you want to draw gizmos that are also pickable and always drawn.
OnDrawGizmosSelected	Implement OnDrawGizmosSelected to draw a gizmo if the object is selected.
OnEnable	This function is called when the object becomes enabled and active.
OnFailedToConnect	Called on the client when a connection attempt fails for some reason.
OnFailedToConnectToMasterServer	Called on clients or servers when there is a problem connecting to the MasterServer.
OnGUI	OnGUI is called for rendering and handling GUI events.
OnJointBreak	Called when a joint attached to the same game object broke.
OnJointBreak2D	Called when a Joint2D attached to the same game object breaks.
OnMasterServerEvent	Called on clients or servers when reporting events from the MasterServer.
OnMouseDown	OnMouseDown is called when the user has pressed the mouse button while over the GUIElement or Collider.
OnMouseDrag	OnMouseDrag is called when the user has clicked on a GUIElement or Collider and is still holding down the mouse.
OnMouseEnter	Called when the mouse enters the GUIElement or Collider.
OnMouseExit	Called when the mouse is not any longer over the GUIElement or Collider.
OnMouseOver	Called every frame while the mouse is over the GUIElement or Collider.
OnMouseUp	OnMouseUp is called when the user has released the mouse button.

OnMouseUpAsButton	OnMouseUpAsButton is only called when the mouse is released over the same GUIElement or Collider as it was pressed.
OnNetworkInstantiate	Called on objects which have been network instantiated with Network.Instantiate.
OnParticleCollision	OnParticleCollision is called when a particle hits a Collider.
OnParticleSystemStopped	OnParticleSystemStopped is called when all particles in the system have died, and no new particles will be born. New particles cease to be created either after Stop is called, or when the duration property of a non-looping system has been exceeded.
OnParticleTrigger	OnParticleTrigger is called when any particles in a particle system meet the conditions in the trigger module.
OnPlayerConnected	Called on the server whenever a new player has successfully connected.
OnPlayerDisconnected	Called on the server whenever a player disconnected from the server.
OnPostRender	OnPostRender is called after a camera finished rendering the Scene.
OnPreCull	OnPreCull is called before a camera culls the Scene.
OnPreRender	OnPreRender is called before a camera starts rendering the Scene.
OnRenderImage	OnRenderImage is called after all rendering is complete to render image.
OnRenderObject	OnRenderObject is called after camera has rendered the Scene.
OnSerializeNetworkView	Used to customize synchronization of variables in a script watched by a network view.
OnServerInitialized	Called on the server whenever a Network.InitializeServer was invoked and has completed.
OnTransformChildrenChanged	This function is called when the list of children of the transform of the GameObject has changed.
OnTransformParentChanged	This function is called when the parent property of the transform of the GameObject has changed.
OnTriggerEnter	OnTriggerEnter is called when the Collider other enters the trigger.
OnTriggerEnter2D	Sent when another object enters a trigger collider attached to this object (2D physics only).
OnTriggerExit	OnTriggerExit is called when the Collider other has stopped touching the trigger.
OnTriggerExit2D	Sent when another object leaves a trigger collider attached to this object (2D physics only).

OnTriggerStay	OnTriggerStay is called once per physics update for every Collider other that is touching the trigger.
OnTriggerStay2D	Sent each frame where another object is within a trigger collider attached to this object (2D physics only).
OnValidate	This function is called when the script is loaded or a value is changed in the inspector (Called in the editor only).
OnWillRenderObject	OnWillRenderObject is called for each camera if the object is visible and not a UI element.
Reset	Reset to default values.
Start	Start is called on the frame when a script is enabled just before any of the Update methods are called the first time.
Update	Update is called every frame, if the MonoBehaviour is enabled.

Description

Information about a mouse button event.

Properties

buttonData	Pointer data associated with the mouse event.
buttonState	The state of the button this frame.

Public Methods

PressedThisFrame	Was the button pressed this frame?
ReleasedThisFrame	Was the button released this frame?

Description

A hit result from a BaseRaycaster.

Properties

depth	The relative depth of the element.
distance	Distance to the hit.
gameObject	The GameObject that was hit by the raycast.
index	Hit index.
isValid	Is there an associated module and a hit GameObject.
module	BaseInputModule that raised the hit.
screenPosition	The screen position from which the raycast was generated.
sortingLayer	The SortingLayer of the hit object.
sortingOrder	The SortingOrder for the hit object.
worldNormal	The normal at the hit location of the raycast.
worldPosition	The world position of the where the raycast has hit.

Public Methods

Clear	Reset the result.

Description

A BaseInputModule designed for mouse / keyboard / controller input.

Input module for working with, mouse, keyboard, or controller.

Properties

cancelButton	Input manager name for the 'cancel' button.
forceModuleActive	Force this module to be active.
horizontalAxis	Input manager name for the horizontal axis button.
inputActionsPerSecond	Number of keyboard / controller inputs allowed per second.
repeatDelay	Delay in seconds before the input actions per second repeat rate takes effect.
submitButton	Maximum number of input events handled per second.
verticalAxis	Input manager name for the vertical axis.

Public Methods

ActivateModule	See BaseInputModule.
DeactivateModule	See BaseInputModule.
IsModuleSupported	See BaseInputModule.
Process	See BaseInputModule.
ShouldActivateModule	See BaseInputModule.
UpdateModule	See BaseInputModule.

Protected Methods

ProcessMouseEvent	Iterate through all the different mouse events.
ProcessMousePress	Calculate and process any mouse button state changes.
ProcessTouchPress	This method is called by Unity whenever a touch event is processed. Override this method with a custom implementation to process touch events yourself.
SendMoveEventToSelectedObject	Calculate and send a move event to the current selected object.

SendSubmitEventToSelectedObject	Calculate and send a submit event to the current selected object.
SendUpdateEventToSelectedObject	Send a update event to the currently selected object.

Inherited Members

Static Properties

kFakeTouchesId	Touch id for when simulating touches on a non touch device.
kMouseLeftId	Id of the cached left mouse pointer event.
kMouseMiddleId	Id of the cached middle mouse pointer event.
kMouseRightId	Id of the cached right mouse pointer event.

Properties

input	The current BaseInput being used by the input module.
inputOverride	Used to override the default BaseInput for the input module.
enabled	Enabled Behaviours are Updated, disabled Behaviours are not.
isActiveAndEnabled	Has the Behaviour had active and enabled called?
gameObject	The game object this component is attached to. A component is always attached to a game object.
tag	The tag of this game object.
transform	The Transform attached to this GameObject.
runInEditMode	Allow a specific instance of a MonoBehaviour to run in edit mode (only available in the editor).
useGUILayout	Disabling this lets you skip the GUI layout phase.
hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.

Public Methods

IsPointerOverGameObject	Is the pointer with the given ID over an EventSystem object?
BroadcastMessage	Calls the method named methodName on every MonoBehaviour in this game object or any of its children.

CompareTag	Is this game object tagged with tag ?
GetComponent	Returns the component of Type type if the game object has one attached, null if it doesn't.
GetComponentInChildren	Returns the component of Type type in the GameObject or any of its children using depth first search.
GetComponentInParent	Returns the component of Type type in the GameObject or any of its parents.
GetComponents	Returns all components of Type type in the GameObject.
GetComponentsInChildren	Returns all components of Type type in the GameObject or any of its children.
GetComponentsInParent	Returns all components of Type type in the GameObject or any of its parents.
SendMessage	Calls the method named methodName on every MonoBehaviour in this game object.
SendMessageUpwards	Calls the method named methodName on every MonoBehaviour in this game object and on every ancestor of the behaviour.
CancelInvoke	Cancels all Invoke calls on this MonoBehaviour.
Invoke	Invokes the method methodName in time seconds.
InvokeRepeating	Invokes the method methodName in time seconds, then repeatedly every repeatRate seconds.
IsInvoking	Is any invoke on methodName pending?
StartCoroutine	Starts a coroutine.
StopAllCoroutines	Stops all coroutines running on this behaviour.
StopCoroutine	Stops the first coroutine named methodName, or the coroutine stored in routine running on this behaviour.
GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.
IsActive	Returns true if the GameObject and the Component are active.
IsDestroyed	Returns true if the native representation of the behaviour has been destroyed.

Protected Methods

GetAxisEventData	Given some input data generate an AxisEventData that can be used by the event system.
GetBaseEventData	Generate a BaseEventData that can be used by the EventSystem.

HandlePointerExitAndEnter	Handle sending enter and exit events when a new enter target is found.
OnDisable	See MonoBehaviour.OnDisable.
OnEnable	See MonoBehaviour.OnEnable.
ClearSelection	Clear all pointers and deselect any selected objects in the EventSystem.
CopyFromTo	Copy one PointerEventData to another.
DeselectIfSelectionChanged	Deselect the current selected GameObject if the currently pointed-at GameObject is different.
GetLastPointerEventData	Return the last PointerEventData for the given touch / mouse id.
GetMousePointerEventData	Return the current MouseState.
GetPointerData	Search the cache for currently active pointers, return true if found.
GetTouchPointerEventData	Given a touch populate the PointerEventData and return if we are pressed or released.
ProcessDrag	Process the drag for the current frame with the given pointer event.
ProcessMove	Process movement for the current frame with the given pointer event.
RemovePointerData	Remove the PointerEventData from the cache.
StateForMouseButton	Given a mouse button return the current state for the frame.
Awake	See MonoBehaviour.Awake.
OnBeforeTransformParentChanged	See MonoBehaviour.OnBeforeTransformParentChanged.
OnCanvasGroupChanged	See MonoBehaviour.OnCanvasGroupChanged.
OnCanvasHierarchyChanged	Called when the state of the parent Canvas is changed.
OnDestroy	See MonoBehaviour.OnDestroy.
OnDidApplyAnimationProperties	See LayoutGroup.OnDidApplyAnimationProperties.
OnDisable	See MonoBehaviour.OnDisable.
OnEnable	See MonoBehaviour.OnEnable.
OnRectTransformDimensionsChange	This callback is called if an associated RectTransform has its dimensions changed.
OnTransformParentChanged	See MonoBehaviour.OnRectTransformParentChanged.
OnValidate	See MonoBehaviour.OnValidate.

Reset	See MonoBehaviour.Reset.
Start	See MonoBehaviour.Start.

Static Methods

DetermineMoveDirection	Given an input movement, determine the best MoveDirection.
FindCommonRoot	Given 2 GameObjects, return a common root GameObject (or null).
FindFirstRaycast	Return the first valid RaycastResult.
print	Logs message to the Unity Console (identical to Debug.Log).
Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Messages

Awake	Awake is called when the script instance is being loaded.
FixedUpdate	This function is called every fixed framerate frame, if the MonoBehaviour is enabled.
LateUpdate	LateUpdate is called every frame, if the Behaviour is enabled.
OnAnimatorIK	Callback for setting up animation IK (inverse kinematics).
OnAnimatorMove	Callback for processing animation movements for modifying root motion.

OnApplicationFocus	Sent to all GameObjects when the player gets or loses focus.
OnApplicationPause	Sent to all GameObjects when the application pauses.
OnApplicationQuit	Sent to all game objects before the application quits.
OnAudioFilterRead	If OnAudioFilterRead is implemented, Unity will insert a custom filter into the audio DSP chain.
OnBecameInvisible	OnBecameInvisible is called when the renderer is no longer visible by any camera.
OnBecameVisible	OnBecameVisible is called when the renderer became visible by any camera.
OnCollisionEnter	OnCollisionEnter is called when this collider/rigidbody has begun touching another rigidbody/collider.
OnCollisionEnter2D	Sent when an incoming collider makes contact with this object's collider (2D physics only).
OnCollisionExit	OnCollisionExit is called when this collider/rigidbody has stopped touching another rigidbody/collider.
OnCollisionExit2D	Sent when a collider on another object stops touching this object's collider (2D physics only).
OnCollisionStay	:ref::OnCollisionStay is called once per frame for every collider/rigidbody that is touching rigidbody/collider.
OnCollisionStay2D	Sent each frame where a collider on another object is touching this object's collider (2D physics only).
OnConnectedToServer	Called on the client when you have successfully connected to a server.
OnControllerColliderHit	OnControllerColliderHit is called when the controller hits a collider while performing a Move.
OnDestroy	Destroying the attached Behaviour will result in the game or Scene receiving OnDestroy.
OnDisable	This function is called when the behaviour becomes disabled.
OnDisconnectedFromServer	Called on the client when the connection was lost or you disconnected from the server.
OnDrawGizmos	Implement OnDrawGizmos if you want to draw gizmos that are also pickable and always drawn.
OnDrawGizmosSelected	Implement OnDrawGizmosSelected to draw a gizmo if the object is selected.
OnEnable	This function is called when the object becomes enabled and active.

OnFailedToConnect	Called on the client when a connection attempt fails for some reason.
OnFailedToConnectToMasterServer	Called on clients or servers when there is a problem connecting to the MasterServer.
OnGUI	OnGUI is called for rendering and handling GUI events.
OnJointBreak	Called when a joint attached to the same game object broke.
OnJointBreak2D	Called when a Joint2D attached to the same game object breaks.
OnMasterServerEvent	Called on clients or servers when reporting events from the MasterServer.
OnMouseDown	OnMouseDown is called when the user has pressed the mouse button while over the GUIElement or Collider.
OnMouseDrag	OnMouseDrag is called when the user has clicked on a GUIElement or Collider and is still holding down the mouse.
OnMouseEnter	Called when the mouse enters the GUIElement or Collider.
OnMouseExit	Called when the mouse is not any longer over the GUIElement or Collider.
OnMouseOver	Called every frame while the mouse is over the GUIElement or Collider.
OnMouseUp	OnMouseUp is called when the user has released the mouse button.
OnMouseUpAsButton	OnMouseUpAsButton is only called when the mouse is released over the same GUIElement or Collider as it was pressed.
OnNetworkInstantiate	Called on objects which have been network instantiated with Network.Instantiate.
OnParticleCollision	OnParticleCollision is called when a particle hits a Collider.
OnParticleSystemStopped	OnParticleSystemStopped is called when all particles in the system have died, and no new particles will be born. New particles cease to be created either after Stop is called, or when the duration property of a non-looping system has been exceeded.
OnParticleTrigger	OnParticleTrigger is called when any particles in a particle system meet the conditions in the trigger module.
OnPlayerConnected	Called on the server whenever a new player has successfully connected.
OnPlayerDisconnected	Called on the server whenever a player disconnected from the server.
OnPostRender	OnPostRender is called after a camera finished rendering the Scene.
OnPreCull	OnPreCull is called before a camera culls the Scene.

OnPreRender	OnPreRender is called before a camera starts rendering the Scene.
OnRenderImage	OnRenderImage is called after all rendering is complete to render image.
OnRenderObject	OnRenderObject is called after camera has rendered the Scene.
OnSerializeNetworkView	Used to customize synchronization of variables in a script watched by a network view.
OnServerInitialized	Called on the server whenever a Network.InitializeServer was invoked and has completed.
OnTransformChildrenChanged	This function is called when the list of children of the transform of the GameObject has changed.
OnTransformParentChanged	This function is called when the parent property of the transform of the GameObject has changed.
OnTriggerEnter	OnTriggerEnter is called when the Collider other enters the trigger.
OnTriggerEnter2D	Sent when another object enters a trigger collider attached to this object (2D physics only).
OnTriggerExit	OnTriggerExit is called when the Collider other has stopped touching the trigger.
OnTriggerExit2D	Sent when another object leaves a trigger collider attached to this object (2D physics only).
OnTriggerStay	OnTriggerStay is called once per physics update for every Collider other that is touching the trigger.
OnTriggerStay2D	Sent each frame where another object is within a trigger collider attached to this object (2D physics only).
OnValidate	This function is called when the script is loaded or a value is changed in the inspector (Called in the editor only).
OnWillRenderObject	OnWillRenderObject is called for each camera if the object is visible and not a UI element.
Reset	Reset to default values.
Start	Start is called on the frame when a script is enabled just before any of the Update methods are called the first time.
Update	Update is called every frame, if the MonoBehaviour is enabled.

Obsolete

Description

Input module used for touch based input.

Inherited Members

Static Properties

kFakeTouchesId	Touch id for when simulating touches on a non touch device.
kMouseLeftId	Id of the cached left mouse pointer event.
kMouseMiddleId	Id of the cached middle mouse pointer event.
kMouseRightId	Id of the cached right mouse pointer event.

Properties

input	The current BaseInput being used by the input module.
inputOverride	Used to override the default BaseInput for the input module.
enabled	Enabled Behaviours are Updated, disabled Behaviours are not.
isActiveAndEnabled	Has the Behaviour had active and enabled called?
gameObject	The game object this component is attached to. A component is always attached to a game object.
tag	The tag of this game object.
transform	The Transform attached to this GameObject.
runInEditMode	Allow a specific instance of a MonoBehaviour to run in edit mode (only available in the editor).
useGUILayout	Disabling this lets you skip the GUI layout phase.
hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.

Public Methods

ActivateModule	Called when the module is activated. Override this if you want custom code to execute when you activate your module.

DeactivateModule	Called when the module is deactivated. Override this if you want custom code to execute when you deactivate your module.
IsModuleSupported	Check to see if the module is supported. Override this if you have a platfrom specific module (eg. TouchInputModule that you do not want to activate on standalone.
IsPointerOverGameObject	Is the pointer with the given ID over an EventSystem object?
Process	Process the current tick for the module.
ShouldActivateModule	Should be activated.
UpdateModule	Update the internal state of the Module.
BroadcastMessage	Calls the method named methodName on every MonoBehaviour in this game object or any of its children.
CompareTag	Is this game object tagged with tag ?
GetComponent	Returns the component of Type type if the game object has one attached, null if it doesn't.
GetComponentInChildren	Returns the component of Type type in the GameObject or any of its children using depth first search.
GetComponentInParent	Returns the component of Type type in the GameObject or any of its parents.
GetComponents	Returns all components of Type type in the GameObject.
GetComponentsInChildren	Returns all components of Type type in the GameObject or any of its children.
GetComponentsInParent	Returns all components of Type type in the GameObject or any of its parents.
SendMessage	Calls the method named methodName on every MonoBehaviour in this game object.
SendMessageUpwards	Calls the method named methodName on every MonoBehaviour in this game object and on every ancestor of the behaviour.
CancelInvoke	Cancels all Invoke calls on this MonoBehaviour.
Invoke	Invokes the method methodName in time seconds.
InvokeRepeating	Invokes the method methodName in time seconds, then repeatedly every repeatRate seconds.
IsInvoking	Is any invoke on methodName pending?
StartCoroutine	Starts a coroutine.
StopAllCoroutines	Stops all coroutines running on this behaviour.

StopCoroutine	Stops the first coroutine named methodName, or the coroutine stored in routine running on this behaviour.
GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.
IsActive	Returns true if the GameObject and the Component are active.
IsDestroyed	Returns true if the native representation of the behaviour has been destroyed.

Protected Methods

GetAxisEventData	Given some input data generate an AxisEventData that can be used by the event system.
GetBaseEventData	Generate a BaseEventData that can be used by the EventSystem.
HandlePointerExitAndEnter	Handle sending enter and exit events when a new enter target is found.
OnDisable	See MonoBehaviour.OnDisable.
OnEnable	See MonoBehaviour.OnEnable.
ClearSelection	Clear all pointers and deselect any selected objects in the EventSystem.
CopyFromTo	Copy one PointerEventData to another.
DeselectIfSelectionChanged	Deselect the current selected GameObject if the currently pointed-at GameObject is different.
GetLastPointerEventData	Return the last PointerEventData for the given touch / mouse id.
GetMousePointerEventData	Return the current MouseState.
GetPointerData	Search the cache for currently active pointers, return true if found.
GetTouchPointerEventData	Given a touch populate the PointerEventData and return if we are pressed or released.
ProcessDrag	Process the drag for the current frame with the given pointer event.
ProcessMove	Process movement for the current frame with the given pointer event.
RemovePointerData	Remove the PointerEventData from the cache.
StateForMouseButton	Given a mouse button return the current state for the frame.
Awake	See MonoBehaviour.Awake.
OnBeforeTransformParentChanged	See MonoBehaviour.OnBeforeTransformParentChanged.

OnCanvasGroupChanged	See MonoBehaviour.OnCanvasGroupChanged.
OnCanvasHierarchyChanged	Called when the state of the parent Canvas is changed.
OnDestroy	See MonoBehaviour.OnDestroy.
OnDidApplyAnimationProperties	See LayoutGroup.OnDidApplyAnimationProperties.
OnDisable	See MonoBehaviour.OnDisable.
OnEnable	See MonoBehaviour.OnEnable.
OnRectTransformDimensionsChange	This callback is called if an associated RectTransform has its dimensions changed.
OnTransformParentChanged	See MonoBehaviour.OnRectTransformParentChanged.
OnValidate	See MonoBehaviour.OnValidate.
Reset	See MonoBehaviour.Reset.
Start	See MonoBehaviour.Start.

Static Methods

DetermineMoveDirection	Given an input movement, determine the best MoveDirection.
FindCommonRoot	Given 2 GameObjects, return a common root GameObject (or null).
FindFirstRaycast	Return the first valid RaycastResult.
print	Logs message to the Unity Console (identical to Debug.Log).
Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.

Operators

bool	Does the object exist?

operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Messages

Awake	Awake is called when the script instance is being loaded.
FixedUpdate	This function is called every fixed framerate frame, if the MonoBehaviour is enabled.
LateUpdate	LateUpdate is called every frame, if the Behaviour is enabled.
OnAnimatorIK	Callback for setting up animation IK (inverse kinematics).
OnAnimatorMove	Callback for processing animation movements for modifying root motion.
OnApplicationFocus	Sent to all GameObjects when the player gets or loses focus.
OnApplicationPause	Sent to all GameObjects when the application pauses.
OnApplicationQuit	Sent to all game objects before the application quits.
OnAudioFilterRead	If OnAudioFilterRead is implemented, Unity will insert a custom filter into the audio DSP chain.
OnBecameInvisible	OnBecameInvisible is called when the renderer is no longer visible by any camera.
OnBecameVisible	OnBecameVisible is called when the renderer became visible by any camera.
OnCollisionEnter	OnCollisionEnter is called when this collider/rigidbody has begun touching another rigidbody/collider.
OnCollisionEnter2D	Sent when an incoming collider makes contact with this object's collider (2D physics only).
OnCollisionExit	OnCollisionExit is called when this collider/rigidbody has stopped touching another rigidbody/collider.
OnCollisionExit2D	Sent when a collider on another object stops touching this object's collider (2D physics only).
OnCollisionStay	:ref::OnCollisionStay is called once per frame for every collider/rigidbody that is touching rigidbody/collider.
OnCollisionStay2D	Sent each frame where a collider on another object is touching this object's collider (2D physics only).
OnConnectedToServer	Called on the client when you have successfully connected to a server.

OnControllerColliderHit	OnControllerColliderHit is called when the controller hits a collider while performing a Move.
OnDestroy	Destroying the attached Behaviour will result in the game or Scene receiving OnDestroy.
OnDisable	This function is called when the behaviour becomes disabled.
OnDisconnectedFromServer	Called on the client when the connection was lost or you disconnected from the server.
OnDrawGizmos	Implement OnDrawGizmos if you want to draw gizmos that are also pickable and always drawn.
OnDrawGizmosSelected	Implement OnDrawGizmosSelected to draw a gizmo if the object is selected.
OnEnable	This function is called when the object becomes enabled and active.
OnFailedToConnect	Called on the client when a connection attempt fails for some reason.
OnFailedToConnectToMasterServer	Called on clients or servers when there is a problem connecting to the MasterServer.
OnGUI	OnGUI is called for rendering and handling GUI events.
OnJointBreak	Called when a joint attached to the same game object broke.
OnJointBreak2D	Called when a Joint2D attached to the same game object breaks.
OnMasterServerEvent	Called on clients or servers when reporting events from the MasterServer.
OnMouseDown	OnMouseDown is called when the user has pressed the mouse button while over the GUIElement or Collider.
OnMouseDrag	OnMouseDrag is called when the user has clicked on a GUIElement or Collider and is still holding down the mouse.
OnMouseEnter	Called when the mouse enters the GUIElement or Collider.
OnMouseExit	Called when the mouse is not any longer over the GUIElement or Collider.
OnMouseOver	Called every frame while the mouse is over the GUIElement or Collider.
OnMouseUp	OnMouseUp is called when the user has released the mouse button.
OnMouseUpAsButton	OnMouseUpAsButton is only called when the mouse is released over the same GUIElement or Collider as it was pressed.
OnNetworkInstantiate	Called on objects which have been network instantiated with

	Network.Instantiate.
OnParticleCollision	OnParticleCollision is called when a particle hits a Collider.
OnParticleSystemStopped	OnParticleSystemStopped is called when all particles in the system have died, and no new particles will be born. New particles cease to be created either after Stop is called, or when the duration property of a non-looping system has been exceeded.
OnParticleTrigger	OnParticleTrigger is called when any particles in a particle system meet the conditions in the trigger module.
OnPlayerConnected	Called on the server whenever a new player has successfully connected.
OnPlayerDisconnected	Called on the server whenever a player disconnected from the server.
OnPostRender	OnPostRender is called after a camera finished rendering the Scene.
OnPreCull	OnPreCull is called before a camera culls the Scene.
OnPreRender	OnPreRender is called before a camera starts rendering the Scene.
OnRenderImage	OnRenderImage is called after all rendering is complete to render image.
OnRenderObject	OnRenderObject is called after camera has rendered the Scene.
OnSerializeNetworkView	Used to customize synchronization of variables in a script watched by a network view.
OnServerInitialized	Called on the server whenever a Network.InitializeServer was invoked and has completed.
OnTransformChildrenChanged	This function is called when the list of children of the transform of the GameObject has changed.
OnTransformParentChanged	This function is called when the parent property of the transform of the GameObject has changed.
OnTriggerEnter	OnTriggerEnter is called when the Collider other enters the trigger.
OnTriggerEnter2D	Sent when another object enters a trigger collider attached to this object (2D physics only).
OnTriggerExit	OnTriggerExit is called when the Collider other has stopped touching the trigger.
OnTriggerExit2D	Sent when another object leaves a trigger collider attached to this object (2D physics only).
OnTriggerStay	OnTriggerStay is called once per physics update for every Collider other that is touching the trigger.

OnTriggerStay2D	Sent each frame where another object is within a trigger collider attached to this object (2D physics only).
OnValidate	This function is called when the script is loaded or a value is changed in the inspector (Called in the editor only).
OnWillRenderObject	OnWillRenderObject is called for each camera if the object is visible and not a UI element.
Reset	Reset to default values.
Start	Start is called on the frame when a script is enabled just before any of the Update methods are called the first time.
Update	Update is called every frame, if the MonoBehaviour is enabled.

Description

Base behaviour that has protected implementations of Unity lifecycle functions.

Public Methods

IsActive	Returns true if the GameObject and the Component are active.
IsDestroyed	Returns true if the native representation of the behaviour has been destroyed.

Protected Methods

Awake	See MonoBehaviour.Awake.
OnBeforeTransformParentChanged	See MonoBehaviour.OnBeforeTransformParentChanged.
OnCanvasGroupChanged	See MonoBehaviour.OnCanvasGroupChanged.
OnCanvasHierarchyChanged	Called when the state of the parent Canvas is changed.
OnDestroy	See MonoBehaviour.OnDestroy.
OnDidApplyAnimationProperties	See LayoutGroup.OnDidApplyAnimationProperties.
OnDisable	See MonoBehaviour.OnDisable.
OnEnable	See MonoBehaviour.OnEnable.
OnRectTransformDimensionsChange	This callback is called if an associated RectTransform has its dimensions changed.
OnTransformParentChanged	See MonoBehaviour.OnRectTransformParentChanged.
OnValidate	See MonoBehaviour.OnValidate.
Reset	See MonoBehaviour.Reset.
Start	See MonoBehaviour.Start.

Inherited Members

Properties

enabled	Enabled Behaviours are Updated, disabled Behaviours are not.
isActiveAndEnabled	Has the Behaviour had active and enabled called?

gameObject	The game object this component is attached to. A component is always attached to a game object.
tag	The tag of this game object.
transform	The Transform attached to this GameObject.
runInEditMode	Allow a specific instance of a MonoBehaviour to run in edit mode (only available in the editor).
useGUILayout	Disabling this lets you skip the GUI layout phase.
hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.

Public Methods

BroadcastMessage	Calls the method named methodName on every MonoBehaviour in this game object or any of its children.
CompareTag	Is this game object tagged with tag ?
GetComponent	Returns the component of Type type if the game object has one attached, null if it doesn't.
GetComponentInChildren	Returns the component of Type type in the GameObject or any of its children using depth first search.
GetComponentInParent	Returns the component of Type type in the GameObject or any of its parents.
GetComponents	Returns all components of Type type in the GameObject.
GetComponentsInChildren	Returns all components of Type type in the GameObject or any of its children.
GetComponentsInParent	Returns all components of Type type in the GameObject or any of its parents.
SendMessage	Calls the method named methodName on every MonoBehaviour in this game object.
SendMessageUpwards	Calls the method named methodName on every MonoBehaviour in this game object and on every ancestor of the behaviour.
CancelInvoke	Cancels all Invoke calls on this MonoBehaviour.
Invoke	Invokes the method methodName in time seconds.
InvokeRepeating	Invokes the method methodName in time seconds, then repeatedly every repeatRate seconds.
IsInvoking	Is any invoke on methodName pending?

StartCoroutine	Starts a coroutine.
StopAllCoroutines	Stops all coroutines running on this behaviour.
StopCoroutine	Stops the first coroutine named methodName, or the coroutine stored in routine running on this behaviour.
GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.

Static Methods

print	Logs message to the Unity Console (identical to Debug.Log).
Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Messages

FixedUpdate	This function is called every fixed framerate frame, if the MonoBehaviour is enabled.
LateUpdate	LateUpdate is called every frame, if the Behaviour is enabled.
OnAnimatorIK	Callback for setting up animation IK (inverse kinematics).
OnAnimatorMove	Callback for processing animation movements for modifying root motion.
OnApplicationFocus	Sent to all GameObjects when the player gets or loses focus.

OnApplicationPause	Sent to all GameObjects when the application pauses.
OnApplicationQuit	Sent to all game objects before the application quits.
OnAudioFilterRead	If OnAudioFilterRead is implemented, Unity will insert a custom filter into the audio DSP chain.
OnBecameInvisible	OnBecameInvisible is called when the renderer is no longer visible by any camera.
OnBecameVisible	OnBecameVisible is called when the renderer became visible by any camera.
OnCollisionEnter	OnCollisionEnter is called when this collider/rigidbody has begun touching another rigidbody/collider.
OnCollisionEnter2D	Sent when an incoming collider makes contact with this object's collider (2D physics only).
OnCollisionExit	OnCollisionExit is called when this collider/rigidbody has stopped touching another rigidbody/collider.
OnCollisionExit2D	Sent when a collider on another object stops touching this object's collider (2D physics only).
OnCollisionStay	:ref::OnCollisionStay is called once per frame for every collider/rigidbody that is touching rigidbody/collider.
OnCollisionStay2D	Sent each frame where a collider on another object is touching this object's collider (2D physics only).
OnConnectedToServer	Called on the client when you have successfully connected to a server.
OnControllerColliderHit	OnControllerColliderHit is called when the controller hits a collider while performing a Move.
OnDisconnectedFromServer	Called on the client when the connection was lost or you disconnected from the server.
OnDrawGizmos	Implement OnDrawGizmos if you want to draw gizmos that are also pickable and always drawn.
OnDrawGizmosSelected	Implement OnDrawGizmosSelected to draw a gizmo if the object is selected.
OnFailedToConnect	Called on the client when a connection attempt fails for some reason.
OnFailedToConnectToMasterServer	Called on clients or servers when there is a problem connecting to the MasterServer.
OnGUI	OnGUI is called for rendering and handling GUI events.
OnJointBreak	Called when a joint attached to the same game object broke.

OnJointBreak2D	Called when a Joint2D attached to the same game object breaks.
OnMasterServerEvent	Called on clients or servers when reporting events from the MasterServer.
OnMouseDown	OnMouseDown is called when the user has pressed the mouse button while over the GUIElement or Collider.
OnMouseDrag	OnMouseDrag is called when the user has clicked on a GUIElement or Collider and is still holding down the mouse.
OnMouseEnter	Called when the mouse enters the GUIElement or Collider.
OnMouseExit	Called when the mouse is not any longer over the GUIElement or Collider.
OnMouseOver	Called every frame while the mouse is over the GUIElement or Collider.
OnMouseUp	OnMouseUp is called when the user has released the mouse button.
OnMouseUpAsButton	OnMouseUpAsButton is only called when the mouse is released over the same GUIElement or Collider as it was pressed.
OnNetworkInstantiate	Called on objects which have been network instantiated with Network.Instantiate.
OnParticleCollision	OnParticleCollision is called when a particle hits a Collider.
OnParticleSystemStopped	OnParticleSystemStopped is called when all particles in the system have died, and no new particles will be born. New particles cease to be created either after Stop is called, or when the duration property of a non-looping system has been exceeded.
OnParticleTrigger	OnParticleTrigger is called when any particles in a particle system meet the conditions in the trigger module.
OnPlayerConnected	Called on the server whenever a new player has successfully connected.
OnPlayerDisconnected	Called on the server whenever a player disconnected from the server.
OnPostRender	OnPostRender is called after a camera finished rendering the Scene.
OnPreCull	OnPreCull is called before a camera culls the Scene.
OnPreRender	OnPreRender is called before a camera starts rendering the Scene.
OnRenderImage	OnRenderImage is called after all rendering is complete to render image.
OnRenderObject	OnRenderObject is called after camera has rendered the Scene.
OnSerializeNetworkView	Used to customize synchronization of variables in a script watched by a

	network view.
OnServerInitialized	Called on the server whenever a Network.InitializeServer was invoked and has completed.
OnTransformChildrenChanged	This function is called when the list of children of the transform of the GameObject has changed.
OnTriggerEnter	OnTriggerEnter is called when the Collider other enters the trigger.
OnTriggerEnter2D	Sent when another object enters a trigger collider attached to this object (2D physics only).
OnTriggerExit	OnTriggerExit is called when the Collider other has stopped touching the trigger.
OnTriggerExit2D	Sent when another object leaves a trigger collider attached to this object (2D physics only).
OnTriggerStay	OnTriggerStay is called once per physics update for every Collider other that is touching the trigger.
OnTriggerStay2D	Sent each frame where another object is within a trigger collider attached to this object (2D physics only).
OnWillRenderObject	OnWillRenderObject is called for each camera if the object is visible and not a UI element.
Update	Update is called every frame, if the MonoBehaviour is enabled.

Implements interfaces:IEventSystemHandler

Description

Interface to implement if you wish to receive OnBeginDrag callbacks.

Note: You need to implement IDragHandler in addition to IBeginDragHandler.

See Also: IDragHandler.

Public Methods

OnBeginDrag	Called by a BaseInputModule before a drag is started.

Implements interfaces:IEventSystemHandler

Description

Interface to implement if you wish to receive OnCancel callbacks.

Public Methods

OnCancel	Called by a BaseInputModule when a Cancel event occurs.

Implements interfaces:IEventSystemHandler

Description

Interface to implement if you wish to receive OnDeselect callbacks.

Public Methods

OnDeselect	Called by the EventSystem when a new object is being selected.

Implements interfaces:IEventSystemHandler

Description

Interface to implement if you wish to receive OnDrag callbacks.

```
using UnityEngine;
using UnityEngine.EventSystems;
using UnityEngine.UI;


[RequireComponent(typeof(Image))]
public class DragMe : MonoBehaviour, IBeginDragHandler, IDragHandler, IEndDragHa
{
    public bool dragOnSurfaces = true;


    private GameObject m_DraggingIcon;
    private RectTransform m_DraggingPlane;


    public void OnBeginDrag(PointerEventData eventData)
    {
        var canvas = FindInParents<Canvas>(gameObject);
        if (canvas == null)
            return;


        // We have clicked something that can be dragged.
        // What we want to do is create an icon for this.
        m_DraggingIcon = new GameObject("icon");


        m_DraggingIcon.transform.SetParent(canvas.transform, false);
        m_DraggingIcon.transform.SetAsLastSibling();


        var image = m_DraggingIcon.AddComponent<Image>();


        image.sprite = GetComponent<Image>().sprite;
        image.SetNativeSize();
```

```
        if (dragOnSurfaces)
            m_DraggingPlane = transform as RectTransform;
        else
            m_DraggingPlane = canvas.transform as RectTransform;


        SetDraggedPosition(eventData);
    }


    public void OnDrag(PointerEventData data)
    {
        if (m_DraggingIcon != null)
            SetDraggedPosition(data);
    }


    private void SetDraggedPosition(PointerEventData data)
    {
        if (dragOnSurfaces && data.pointerEnter != null && data.pointerEnter.tra
            m_DraggingPlane = data.pointerEnter.transform as RectTransform;


        var rt = m_DraggingIcon.GetComponent<RectTransform>();
        Vector3 globalMousePos;
        if (RectTransformUtility.ScreenPointToWorldPointInRectangle(m_DraggingPl
        {
            rt.position = globalMousePos;
            rt.rotation = m_DraggingPlane.rotation;
        }
    }


    public void OnEndDrag(PointerEventData eventData)
    {
        if (m_DraggingIcon != null)
            Destroy(m_DraggingIcon);
    }


    static public T FindInParents<T>(GameObject go) where T : Component
    {
        if (go == null) return null;
        var comp = go.GetComponent<T>();
```

```
        if (comp != null)
            return comp;


        Transform t = go.transform.parent;
        while (t != null && comp == null)
        {
            comp = t.gameObject.GetComponent<T>();
            t = t.parent;
        }
        return comp;
    }
}
```

Public Methods

OnDrag	When draging is occuring this will be called every time the cursor is moved.

Implements interfaces:IEventSystemHandler

Description

Interface to implement if you wish to receive OnDrop callbacks.

```
using UnityEngine;
using UnityEngine.EventSystems;


public class Example : MonoBehaviour, IDropHandler
{
    public void OnDrop(PointerEventData data)
    {
        if (data.pointerDrag != null)
        {
            Debug.Log("Dropped object was: " + data.pointerDrag);
        }
    }
}
```

Public Methods

OnDrop	Called by a BaseInputModule on a target that can accept a drop.

Implements interfaces:IEventSystemHandler

Description

Interface to implement if you wish to receive OnEndDrag callbacks.

Note: You need to implement IDragHandler in addition to IEndDragHandler.

See Also: IDragHandler.

Public Methods

OnEndDrag	Called by a BaseInputModule when a drag is ended.

Description

Base class that all EventSystem events inherit from.

Implements interfaces:IEventSystemHandler

Description

Interface to implement if you wish to receive OnInitializePotentialDrag callbacks.

Public Methods

OnInitializePotentialDrag	Called by a BaseInputModule when a drag has been found but before it is valid to begin the drag.

Implements interfaces:IEventSystemHandler

Description

Interface to implement if you wish to receive OnMove callbacks.

Public Methods

OnMove	Called by a BaseInputModule when a move event occurs.

Implements interfaces:IEventSystemHandler

Description

Interface to implement if you wish to receive OnPointerClick callbacks.

Use the IPointerClickHandler Interface to handle click input using OnPointerClick callbacks. Ensure an Event System exists in the Scene to allow click detection. For click detection on non-UI GameObjects, ensure a PhysicsRaycaster is attached to the Camera.

```
using UnityEngine;
using UnityEngine.EventSystems;


public class Example : MonoBehaviour, IPointerClickHandler
{
    //Detect if a click occurs
    public void OnPointerClick(PointerEventData pointerEventData)
    {
        //Output to console the clicked GameObject's name and the following mess
        Debug.Log(name + " Game Object Clicked!");
    }
}
```

Public Methods

OnPointerClick	Use this callback to detect clicks.

Implements interfaces:IEventSystemHandler

Description

Interface to implement if you wish to receive OnPointerDown callbacks.

Detects ongoing mouse clicks until release of the mouse button. Use IPointerUpHandler to handle the release of the mouse button.

```
//Attach this script to the GameObject you would like to have mouse clicks detec
//This script outputs a message to the Console when a click is currently detecte


using UnityEngine;
using UnityEngine.EventSystems;


public class Example : MonoBehaviour, IPointerDownHandler, IPointerUpHandler
{
    //Detect current clicks on the GameObject (the one with the script attached)
    public void OnPointerDown(PointerEventData pointerEventData)
    {
        //Output the name of the GameObject that is being clicked
        Debug.Log(name + "Game Object Click in Progress");
    }


    //Detect if clicks are no longer registering
    public void OnPointerUp(PointerEventData pointerEventData)
    {
        Debug.Log(name + "No longer being clicked");
    }
}
```

Public Methods

OnPointerDown

Implements interfaces:IEventSystemHandler

Description

Interface to implement if you wish to receive OnPointerEnter callbacks.

Use to detect when the mouse begins to hover over a certain GameObject. To detect when the mouse stops hovering over the GameObject, use IPointerExitHandler.

```
//Attach this script to the GameObject you would like to have mouse hovering det
//This script outputs a message to the Console when the mouse pointer is current


using UnityEngine;
using UnityEngine.EventSystems;


public class Example : MonoBehaviour, IPointerEnterHandler, IPointerExitHandler
{
    //Detect if the Cursor starts to pass over the GameObject
    public void OnPointerEnter(PointerEventData pointerEventData)
    {
        //Output to console the GameObject's name and the following message
        Debug.Log("Cursor Entering " + name + " GameObject");
    }


    //Detect when Cursor leaves the GameObject
    public void OnPointerExit(PointerEventData pointerEventData)
    {
        //Output the following message with the GameObject's name
        Debug.Log("Cursor Exiting " + name + " GameObject");
    }
}
```

Public Methods

OnPointerEnter

Implements interfaces:IEventSystemHandler

Description

Interface to implement if you wish to receive OnPointerExit callbacks.

Public Methods

OnPointerExit

Implements interfaces:IEventSystemHandler

Description

Interface to implement if you wish to receive OnPointerUp callbacks.

Note: In order to receive OnPointerUp callbacks, you must also implement the IPointerDownHandler interface

Public Methods

OnPointerUp

Implements interfaces:IEventSystemHandler

Description

Interface to implement if you wish to receive OnScroll callbacks.

Public Methods

OnScroll

Implements interfaces:IEventSystemHandler

Description

Interface to implement if you wish to receive OnSelect callbacks.

Public Methods

OnSelect

Implements interfaces:IEventSystemHandler

Description

Interface to implement if you wish to receive OnSubmit callbacks.

Public Methods

OnSubmit

Implements interfaces:IEventSystemHandler

Description

Interface to implement if you wish to receive OnUpdateSelected callbacks.

Public Methods

OnUpdateSelected	Called by the EventSystem when the object associated with this EventTrigger is updated.

Description

Enum that tracks event State.

Properties

Unused	Unused.
Used	Used.

Description

The type of event the TriggerEvent is intercepting.

Properties

PointerEnter	Intercepts a IPointerEnterHandler.OnPointerEnter.
PointerExit	Intercepts a IPointerExitHandler.OnPointerExit.
PointerDown	Intercepts a IPointerDownHandler.OnPointerDown.
PointerUp	Intercepts a IPointerUpHandler.OnPointerUp.
PointerClick	Intercepts a IPointerClickHandler.OnPointerClick.
Drag	Intercepts a IDragHandler.OnDrag.
Drop	Intercepts a IDropHandler.OnDrop.
Scroll	Intercepts a IScrollHandler.OnScroll.
UpdateSelected	Intercepts a IUpdateSelectedHandler.OnUpdateSelected.
Select	Intercepts a ISelectHandler.OnSelect.
Deselect	Intercepts a IDeselectHandler.OnDeselect.
Move	Intercepts a IMoveHandler.OnMove.
InitializePotentialDrag	Intercepts IInitializePotentialDrag.InitializePotentialDrag.
BeginDrag	Intercepts IBeginDragHandler.OnBeginDrag.
EndDrag	Intercepts IEndDragHandler.OnEndDrag.
Submit	Intercepts ISubmitHandler.Submit.
Cancel	Intercepts ICancelHandler.OnCancel.

Description

This is an 8 direction movement enum.

MoveDirection provides a way of switching between moving states. You must assign these states to actions, such as moving the GameObject by an up vector when in the Up state.

Having states like these are easier to identify than always having to include a large amount of vectors and calculations. Instead, you define what you want the state to do in only one part, and switch to the appropriate state when it is needed.

```
//This is a full example of how a GameObject changes direction using MoveDirecti
//Assign this script to a visible GameObject (with a Rigidbody attached) to see


using UnityEngine;
using UnityEngine.EventSystems;


public class Example : MonoBehaviour
{
    Vector3 m_StartPosition, m_StartForce;
    Rigidbody m_Rigidbody;
    //Use Enum for easy switching between direction states
    MoveDirection m_MoveDirection;


    //Use these Vectors for moving Rigidbody components
    Vector3 m_ResetVector;
    Vector3 m_UpVector;
    Vector3 m_RightVector;
    const float speed = 5.0f;


    void Start()
    {
        //You get the Rigidbody component attached to the GameObject
        m_Rigidbody = GetComponent<Rigidbody>();
        //This starts with the Rigidbody not moving in any direction at all
        m_MoveDirection = MoveDirection.None;


        //These are the GameObject's starting position and Rigidbody position
```

```
        m_StartPosition = transform.position;
        m_StartForce = m_Rigidbody.transform.position;


        //This Vector is set to 1 in the y axis (for moving upwards)
        m_UpVector = Vector3.up;
        //This Vector is set to 1 in the x axis (for moving in the right directi
        m_RightVector = Vector3.right;
        //This Vector is zeroed out for when the Rigidbody should not move
        m_ResetVector = Vector3.zero;
    }


    void Update()
    {
        //This switches the direction depending on button presses
        switch (m_MoveDirection)
        {
            //The starting state which resets the object
            case MoveDirection.None:
                //Reset to the starting position of the GameObject and Rigidbody
                transform.position = m_StartPosition;
                m_Rigidbody.transform.position = m_StartForce;
                //This resets the velocity of the Rigidbody
                m_Rigidbody.velocity = m_ResetVector;
                break;


            //This is for moving in an upwards direction
            case MoveDirection.Up:
                //Change the velocity so that the Rigidbody travels upwards
                m_Rigidbody.velocity = m_UpVector * speed;
                break;


            //This is for moving left
            case MoveDirection.Left:
                //This moves the Rigidbody to the left (minus right Vector)
                m_Rigidbody.velocity = -m_RightVector * speed;
                break;


            //This is for moving right
            case MoveDirection.Right:
                //This moves the Rigidbody to the right
                m_Rigidbody.velocity = m_RightVector * speed;
```

```
            break;


        //This is for moving down
        case MoveDirection.Down:
            //This moves the Rigidbody down
            m_Rigidbody.velocity = -m_UpVector * speed;
            break;
    }
}


void OnGUI()
{
    //Press the reset Button to switch to no mode
    if (GUI.Button(new Rect(100, 0, 150, 30), "Reset"))
    {
        //Switch to start/reset case
        m_MoveDirection = MoveDirection.None;
    }


    //Press the Left button to switch the Rigidbody direction to the left
    if (GUI.Button(new Rect(100, 30, 150, 30), "Move Left"))
    {
        //Switch to the left direction
        m_MoveDirection = MoveDirection.Left;
    }


    //Press the Up button to switch the Rigidbody direction to upwards
    if (GUI.Button(new Rect(100, 60, 150, 30), "Move Up"))
    {
        //Switch to Up Direction
        m_MoveDirection = MoveDirection.Up;
    }


    //Press the Down button to switch the direction to down
    if (GUI.Button(new Rect(100, 90, 150, 30), "Move Down"))
    {
        //Switch to Down Direction
        m_MoveDirection = MoveDirection.Down;
    }
```

```
        //Press the right button to switch to the right direction
        if (GUI.Button(new Rect(100, 120, 150, 30), "Move Right"))
        {
            //Switch to Right Direction
            m_MoveDirection = MoveDirection.Right;
        }
    }
}
```

Properties

Left	This is the Left state of MoveDirection. Assign functionality for moving to the left.
Up	This is the Up state of MoveDirection. Assign functionality for moving in an upward direction.
Right	This is the Right state of MoveDirection. Assign functionality for moving to the right.
Down	The Down State of MoveDirection. Assign functionality for moving in a downward direction.
None	This is the None state. Assign functionality that stops movement.

Description

A world position that is guaranteed to be on the surface of the NavMesh.

The NavMeshLocation stores the position on the NavMesh surface together with the PolygonId of the NavMesh node containing that position. Using NavMeshLocations with NavMeshQuery operations remove the need to project the desired world position onto the NavMesh at the beginning of each and every operation.

A NavMeshLocation can be invalid in two situations: 1. When it has been created empty, instead of being the result of a NavMeshQuery operation. 2. When the NavMesh has been removed or modified at the indicated position or in its close vicinity.

If a NavMeshLocation is made invalid by a NavMeshObstacle carving the NavMesh in its vicinity the NavMeshLocation returns to a valid state if the NavMeshObstacle is removed. This is because removing a NavMeshObstacle restores the NavMesh to its original form without regenerating it.

See Also: NavMeshQuery.MapLocation, NavMeshQuery.IsValid, PolygonId.

Properties

polygon	Unique identifier for the node in the NavMesh to which the world position has been mapped.
position	A world position that sits precisely on the surface of the NavMesh or along its links.

Description

Object used for doing navigation operations in a NavMeshWorld.

NavMeshQuery operations can be executed inside jobs (IJob, IJobParallelFor), as opposed to the operations in the NavMesh-related structures.

To obtain a path between two locations on the NavMesh, the query must have been created with a `pathNodePoolSize` value greater than 0 and the following methods need to be called in sequence: `BeginFindPath`, `UpdateFindPath` (repeatedly, if necessary), `EndFindPath`, `GetPathResult`. These methods rely on storing some state data inside the NavMeshQuery but all the other methods do not change the object's state and can be called any time.

All methods throw exceptions if any of their parameters are not valid when executed in the Editor.

Note: The intended feature set for NavMeshQuery is not yet fully complete.

Constructors

NavMeshQuery	Creates the NavMeshQuery object and allocates memory to store NavMesh node information, if required.

Public Methods

BeginFindPath	Initiates a pathfinding operation between two locations on the NavMesh.
CreateLocation	Returns a valid NavMeshLocation for a position and a polygon provided by the user.
Dispose	Destroys the NavMeshQuery and deallocates all memory used by it.
EndFindPath	Obtains the number of nodes in the path that has been computed during a successful NavMeshQuery.UpdateFindPath operation.
GetAgentTypeIdForPolygon	Returns the identifier of the agent type the NavMesh was baked for or for which the link has been configured.
GetPathResult	Copies into the provided array the list of NavMesh nodes that form the path found by the NavMeshQuery operation.
GetPolygonType	Returns whether the NavMesh node is a polygon or a link.
GetPortalPoints	Obtains the end points of the line segment common to two adjacent NavMesh nodes.

IsValid	Returns true if the node referenced by the specified PolygonId is active in the NavMesh.
MapLocation	Finds the closest point and PolygonId on the NavMesh for a given world position.
MoveLocation	Translates a NavMesh location to another position without losing contact with the surface.
MoveLocations	Translates a series of NavMesh locations to other positions without losing contact with the surface.
MoveLocationsInSameAreas	Translates a series of NavMesh locations to other positions without losing contact with the surface, given one common area filter for all of them.
PolygonLocalToWorldMatrix	Returns the transformation matrix of the NavMesh surface that contains the specified NavMesh node (Read Only).
PolygonWorldToLocalMatrix	Returns the inverse transformation matrix of the NavMesh surface that contains the specified NavMesh node (Read Only).
Raycast	Trace a line between two points on the NavMesh.
UpdateFindPath	Continues a path search that is in progress.

Description

Assembles together a collection of NavMesh surfaces and links that are used as a whole for performing navigation operations.

Operations are initialized against one world, can use only the NavMeshes inside that world and are not aware of the existence of any other NavMeshWorld.

Copying this object only produces a new reference to the same NavMesh data, it does not duplicate the data in memory.

Important note: Currently only a single NavMesh world can be used and a reference to it can be obtained through the NavMeshWorld.GetDefaultWorld method. In the future, multiple NavMesh worlds will be able to be created and any two of them will be completely isolated from each other.

See Also: NavMeshQuery.

Public Methods

AddDependency	Tells the NavMesh world to halt any changes until the specified job is completed.
IsValid	Returns true if the NavMeshWorld has been properly initialized.

Static Methods

GetDefaultWorld	Returns a reference to the single NavMeshWorld that can currently exist and be used in Unity.

Description

Represents a compact identifier for the data of a NavMesh node.

It is used in NavMeshQuery operations for pinpointing and getting access to relevant nodes in the NavMesh. Each node can be used by only one type of agent. See Also: NavMeshSurface.

This identifier becomes invalid once the node gets removed from the NavMesh, either by completely removing the surface or by modifying the surface in the node's immediate vicinity.

See Also: NavMeshQuery.IsValid.

Public Methods

Equals	Returns true if two PolygonId objects refer to the same NavMesh node.
GetHashCode	Returns the hash code for use in collections.
IsNull	Returns true if the PolygonId has been created empty and has never pointed to any node in the NavMesh.

Operators

operator !=	Returns true if two PolygonId objects refer to different NavMesh nodes or if only one of them is null.
operator ==	Returns true if two PolygonId objects refer to the same NavMesh node or if they are both null.

Description

The types of nodes in the navigation data.

Navigation data is comprised geometrically of polygons and segments connected together.

See Also: NavMeshQuery.GetPolygonType, NavMeshSurface, NavMeshLink, Off-mesh Link.

Properties

Ground	Type of node in the NavMesh representing one surface polygon.
OffMeshConnection	Type of node in the NavMesh representing a point-to-point connection between two positions on the NavMesh surface.

Description

Bit flags representing the resulting state of NavMeshQuery operations.

The main values are `Success`, `Failure` and `InProgress`. A status will usually have only one of these main flags set. The secondary flags (details) are set when specific issues have been encountered during the operation. `StatusDetailMask` is a bit mask that can be used to filter out these secondary flags.

Note: Issues highlighted by the presence of certain detail flags in certain situations might refer to internal structures outside the control of users, thus they will not always be able to mitigate them by taking the necessary actions in their code. Ways for handling these situations will be made available in the future.

Properties

Failure	The operation has failed.
Success	The operation was successful.
InProgress	The operation is in progress.
StatusDetailMask	Bitmask that has 0 set for the Success, Failure and InProgress bits and 1 set for all the other flags.
WrongMagic	Data in the NavMesh cannot be recognized and used.
WrongVersion	Data in the NavMesh world has a wrong version.
OutOfMemory	Operation ran out of memory.
InvalidParam	A parameter did not contain valid information, useful for carring out the NavMesh query.
BufferTooSmall	The node buffer of the query was too small to store all results.
OutOfNodes	Query ran out of node stack space during a search.
PartialResult	Query did not reach the end location, returning best guess.

Description

The humanoid stream of animation data passed from one Playable to another.

The AnimationHumanStream structure is passed through the animation Playable structures, like AnimationClipPlayable and AnimationMixerPlayable. They can be modified when used with an IAnimationJobPlayable, like the AnimationScriptPlayable.

The Playables implementing IAnimationJobPlayable take a custom C# job, which must implement IAnimationJob, and the AnimationHumanStream is then passed to its callbacks during the animation processing pass.

See Also: AnimationStream, AnimationStream.isHumanStream, and AnimationStream.AsHuman().

Properties

bodyLocalPosition	The position of the body center of mass relative to the root.
bodyLocalRotation	The rotation of the body center of mass relative to the root.
bodyPosition	The position of the body center of mass in world space.
bodyRotation	The rotation of the body center of mass in world space.
humanScale	The scale of the Avatar. (Read Only)
isValid	Returns true if the stream is valid; false otherwise. (Read Only)
leftFootHeight	The left foot height from the floor. (Read Only)
leftFootVelocity	The left foot velocity from the last evaluated frame. (Read Only)
rightFootHeight	The right foot height from the floor. (Read Only)
rightFootVelocity	The right foot velocity from the last evaluated frame. (Read Only)

Public Methods

GetGoalLocalPosition	Returns the position of this IK goal relative to the root.
GetGoalLocalRotation	Returns the rotation of this IK goal relative to the root.
GetGoalPosition	Returns the position of this IK goal in world space.
GetGoalPositionFromPose	Returns the position of this IK goal in world space computed from the stream current pose.
GetGoalRotation	Returns the rotation of this IK goal in world space.

GetGoalRotationFromPose	Returns the rotation of this IK goal in world space computed from the stream current pose.
GetGoalWeightPosition	Returns the position weight of the IK goal.
GetGoalWeightRotation	Returns the rotation weight of the IK goal.
GetHintPosition	Returns the position of this IK Hint in world space.
GetHintWeightPosition	Returns the position weight of the IK Hint.
GetMuscle	Returns the muscle value.
ResetToStancePose	Reset the current pose to the stance pose (T Pose).
SetGoalLocalPosition	Sets the position of this IK goal relative to the root.
SetGoalLocalRotation	Sets the rotation of this IK goal relative to the root.
SetGoalPosition	Sets the position of this IK goal in world space.
SetGoalRotation	Sets the rotation of this IK goal in world space.
SetGoalWeightPosition	Sets the position weight of the IK goal.
SetGoalWeightRotation	Sets the rotation weight of the IK goal.
SetHintPosition	Sets the position of this IK hint in world space.
SetHintWeightPosition	Sets the position weight of the IK Hint.
SetLookAtBodyWeight	Sets the LookAt body weight.
SetLookAtClampWeight	Sets the LookAt clamp weight.
SetLookAtEyesWeight	Sets the LookAt eyes weight.
SetLookAtHeadWeight	Sets the LookAt head weight.
SetLookAtPosition	Sets the look at position in world space.
SetMuscle	Sets the muscle value.
SolveIK	Execute the IK solver.

Implements interfaces:IAnimationJobPlayable, IPlayable

Description

A Playable that can run a custom, multi-threaded animation job.

This playable allows to create a custom C# job that will give read and write access to the AnimationStream during the animation process pass in the PlayableGraph. The C# job must implement the interface IAnimationJob.

NOTE: You can use PlayableExtensions methods with AnimationScriptPlayable objects.

See Also: IAnimationJob, and AnimationScriptPlayable.Create.

Public Methods

GetJobData	Gets the job data contained in the playable.
GetProcessInputs	Returns whether the playable inputs will be processed or not.
SetJobData	Sets a new job data in the playable.
SetProcessInputs	Sets the new value for processing the inputs or not.

Static Methods

Create	Creates an AnimationScriptPlayable in the PlayableGraph.

Description

The stream of animation data passed from one Playable to another.

The AnimationStream structure is passed through the animation Playable structures, like AnimationClipPlayable and AnimationMixerPlayable. They can be modified when used with an IAnimationJobPlayable, like the AnimationScriptPlayable.

The Playables implementing IAnimationJobPlayable take a custom C# job, which must implement IAnimationJob, and the AnimationStream is then passed to its callbacks during the animation processing pass.

See Also: IAnimationJob, AnimationScriptPlayable, TransformStreamHandle, PropertyStreamHandle, TransformSceneHandle, and PropertySceneHandle.

Properties

angularVelocity	Gets or sets the avatar angular velocity for the evaluated frame.
deltaTime	Gets the delta time for the evaluated frame. (Read Only)
inputStreamCount	Gets the number of input streams. (Read Only)
isHumanStream	Returns true if the stream is from a humanoid avatar; false otherwise. (Read Only)
isValid	Returns true if the stream is valid; false otherwise. (Read Only)
rootMotionPosition	Gets the root motion position for the evaluated frame. (Read Only)
rootMotionRotation	Gets the root motion rotation for the evaluated frame. (Read Only)
velocity	Gets or sets the avatar velocity for the evaluated frame.

Public Methods

AsHuman	Gets the same stream, but as an AnimationHumanStream.
GetInputStream	Gets the AnimationStream of the playable input at index.

Description

Static class providing extension methods for Animator and the animation C# jobs.

The extension methods in this class can directly be used on an Animator.

See Also: IAnimationJobPlayable.

Static Methods

BindSceneProperty	Create a PropertySceneHandle representing the new binding on the Component property of a Transform in the Scene.
BindSceneTransform	Create a TransformSceneHandle representing the new binding between the Animator and a Transform in the Scene.
BindStreamProperty	Create a PropertyStreamHandle representing the new binding on the Component property of a Transform already bound to the Animator.
BindStreamTransform	Create a TransformStreamHandle representing the new binding between the Animator and a Transform already bound to the Animator.
CloseAnimationStream	Close a stream that has been opened using OpenAnimationStream.
OpenAnimationStream	Open a new stream on the Animator.
ResolveAllSceneHandles	Newly created handles are always resolved lazily on the next access when the jobs are run. To avoid a cpu spike while evaluating the jobs you can manually resolve all handles from the main thread.
ResolveAllStreamHandles	Newly created handles are always resolved lazily on the next access when the jobs are run. To avoid a cpu spike while evaluating the jobs you can manually resolve all handles from the main thread.

Description

Handle for a muscle in the AnimationHumanStream.

MuscleHandle can only be used on AnimationHumanStream, otherwise an `InvalidOperationException` is thrown.

```
using UnityEngine;
using UnityEngine.Animations;
using UnityEngine.Playables;
using UnityEngine.Experimental.Animations;


public struct MuscleHandleExampleJob : IAnimationJob
{
    public MuscleHandle muscleHandle;


    public void ProcessRootMotion(AnimationStream stream) {}
    public void ProcessAnimation(AnimationStream stream)
    {
        AnimationHumanStream humanStream = stream.AsHuman();


        // Get a muscle value.
        float muscleValue = humanStream.GetMuscle(muscleHandle);


        // Set a muscle value.
        humanStream.SetMuscle(muscleHandle, muscleValue);
    }
}


[RequireComponent(typeof(Animator))]
public class MuscleHandleExample : MonoBehaviour
{
    void Start()
    {
        var graph = PlayableGraph.Create();
        var output = AnimationPlayableOutput.Create(graph, "output", GetComponen
```

```
        var job = new MuscleHandleExampleJob();
        job.muscleHandle = new MuscleHandle(HumanPartDof.LeftArm, ArmDof.HandDow


        var scriptPlayable = AnimationScriptPlayable.Create(graph, job);
        output.SetSourcePlayable(scriptPlayable);


        graph.Evaluate(1.0f);


        graph.Destroy();
    }
}
```

Static Properties

muscleHandleCount	The total number of DoF parts in a humanoid. (Read Only)

Properties

dof	The muscle human sub-part. (Read Only)
humanPartDof	The muscle human part. (Read Only)
name	The name of the muscle. (Read Only)

Constructors

MuscleHandle	The different constructors that creates the muscle handle.

Static Methods

GetMuscleHandles	Fills the array with all the possible muscle handles on a humanoid.

Description

Handle for a Component property on an object in the Scene.

```
using UnityEngine;
using UnityEngine.Playables;
using UnityEngine.Animations;


using UnityEngine.Experimental.Animations;


public struct PropertySceneHandleJob : IAnimationJob
{
    public PropertySceneHandle handleR;
    public PropertySceneHandle handleG;
    public PropertySceneHandle handleB;
    public Color color;


    public void ProcessRootMotion(AnimationStream stream)
    {
    }


    public void ProcessAnimation(AnimationStream stream)
    {
        // Set the new light color.
        handleR.SetFloat(stream, color.r);
        handleG.SetFloat(stream, color.g);
        handleB.SetFloat(stream, color.b);
    }
}


[RequireComponent(typeof(Animator))]
[RequireComponent(typeof(Light))]
public class PropertySceneHandleExample : MonoBehaviour
{
    public Light sceneLight;
    public Color color = Color.white;
```

```
    PlayableGraph m_Graph;
    AnimationScriptPlayable m_AnimationScriptPlayable;


    void Start()
    {
        if (sceneLight == null)
            return;


        var animator = GetComponent<Animator>();


        m_Graph = PlayableGraph.Create("PropertySceneHandleExample");
        var output = AnimationPlayableOutput.Create(m_Graph, "output", animator)


        var animationJob = new PropertySceneHandleJob();
        animationJob.handleR = animator.BindSceneProperty(sceneLight.transform,
        animationJob.handleG = animator.BindSceneProperty(sceneLight.transform,
        animationJob.handleB = animator.BindSceneProperty(sceneLight.transform,
        m_AnimationScriptPlayable = AnimationScriptPlayable.Create(m_Graph, anim


        output.SetSourcePlayable(m_AnimationScriptPlayable);
        m_Graph.Play();
    }


    void Update()
    {
        if (sceneLight == null)
            return;


        var animationJob = m_AnimationScriptPlayable.GetJobData<PropertySceneHan
        animationJob.color = color;
        m_AnimationScriptPlayable.SetJobData(animationJob);
    }


    void OnDisable()
    {
        if (sceneLight == null)
            return;
```

```
        m_Graph.Destroy();
    }
}
```

See Also: AnimatorJobExtensions.BindSceneProperty, TransformSceneHandle, PropertyStreamHandle, and TransformStreamHandle.

Public Methods

GetBool	Gets the boolean property value from an object in the Scene.
GetFloat	Gets the float property value from an object in the Scene.
GetInt	Gets the integer property value from an object in the Scene.
IsResolved	Returns whether or not the handle is resolved.
IsValid	Returns whether or not the handle is valid.
Resolve	Resolves the handle.
SetBool	Sets the boolean property value to an object in the Scene.
SetFloat	Sets the float property value to an object in the Scene.
SetInt	Sets the integer property value to an object in the Scene.

Description

Handle for a Component property on an object in the AnimationStream.

```
using UnityEngine;
using UnityEngine.Playables;
using UnityEngine.Animations;


using UnityEngine.Experimental.Animations;


public struct PropertyStreamHandleJob : IAnimationJob
{
    public PropertyStreamHandle handleR;
    public PropertyStreamHandle handleG;
    public PropertyStreamHandle handleB;
    public Color color;


    public void ProcessRootMotion(AnimationStream stream)
    {
    }


    public void ProcessAnimation(AnimationStream stream)
    {
        // Set the new light color.
        handleR.SetFloat(stream, color.r);
        handleG.SetFloat(stream, color.g);
        handleB.SetFloat(stream, color.b);
    }
}


[RequireComponent(typeof(Animator))]
[RequireComponent(typeof(Light))]
public class PropertyStreamHandleExample : MonoBehaviour
{
    public Color color = Color.white;


    PlayableGraph m_Graph;
```

```
    AnimationScriptPlayable m_AnimationScriptPlayable;


    void Start()
    {
        var animator = GetComponent<Animator>();


        m_Graph = PlayableGraph.Create("PropertyStreamHandleExample");
        var output = AnimationPlayableOutput.Create(m_Graph, "output", animator)


        var animationJob = new PropertyStreamHandleJob();
        animationJob.handleR = animator.BindStreamProperty(gameObject.transform,
        animationJob.handleG = animator.BindStreamProperty(gameObject.transform,
        animationJob.handleB = animator.BindStreamProperty(gameObject.transform,
        m_AnimationScriptPlayable = AnimationScriptPlayable.Create(m_Graph, anim


        output.SetSourcePlayable(m_AnimationScriptPlayable);
        m_Graph.Play();
    }


    void Update()
    {
        var animationJob = m_AnimationScriptPlayable.GetJobData<PropertyStreamHa
        animationJob.color = color;
        m_AnimationScriptPlayable.SetJobData(animationJob);
    }


    void OnDisable()
    {
        m_Graph.Destroy();
    }
}
```

See Also: AnimatorJobExtensions.BindStreamProperty, TransformStreamHandle, PropertySceneHandle, and TransformSceneHandle.

Public Methods

GetBool	Gets the boolean property value from a stream.

GetFloat	Gets the float property value from a stream.
GetInt	Gets the integer property value from a stream.
IsResolved	Returns whether or not the handle is resolved.
IsValid	Returns whether or not the handle is valid.
Resolve	Resolves the handle.
SetBool	Sets the boolean property value into a stream.
SetFloat	Sets the float property value into a stream.
SetInt	Sets the integer property value into a stream.

Description

Position, rotation and scale of an object in the Scene.

A TransformSceneHandle is a safe handle on a TransformAccess. The Animator used to create this handle manages the validity of this handle.

```
using UnityEngine;
using UnityEngine.Playables;
using UnityEngine.Animations;


using UnityEngine.Experimental.Animations;


public struct TransformSceneHandleJob : IAnimationJob
{
    public TransformSceneHandle handle;
    public Vector3 position;
    public Vector3 rotation;
    public Vector3 scale;


    public void ProcessRootMotion(AnimationStream stream)
    {
        // Set the new local position.
        handle.SetLocalPosition(stream, position);


        // Set the new local rotation (converted from euler).
        handle.SetLocalRotation(stream, Quaternion.Euler(rotation));


        // Set the new local scale.
        handle.SetLocalScale(stream, scale);
    }


    public void ProcessAnimation(AnimationStream stream)
    {
    }
}
```

```
[RequireComponent(typeof(Animator))]
public class TransformSceneHandleExample : MonoBehaviour
{
    public Transform sceneTransform;
    public Vector3 position;
    public Vector3 rotation;
    public Vector3 scale = Vector3.one;


    PlayableGraph m_Graph;
    AnimationScriptPlayable m_AnimationScriptPlayable;


    void Start()
    {
        if (sceneTransform == null)
            return;


        var animator = GetComponent<Animator>();


        m_Graph = PlayableGraph.Create("TransformSceneHandleExample");
        var output = AnimationPlayableOutput.Create(m_Graph, "output", animator)


        var animationJob = new TransformSceneHandleJob();
        animationJob.handle = animator.BindSceneTransform(sceneTransform);
        m_AnimationScriptPlayable = AnimationScriptPlayable.Create(m_Graph, anim


        output.SetSourcePlayable(m_AnimationScriptPlayable);
        m_Graph.Play();
    }


    void Update()
    {
        if (sceneTransform == null)
            return;


        var animationJob = m_AnimationScriptPlayable.GetJobData<TransformSceneHa
        animationJob.position = position;
        animationJob.rotation = rotation;
```

```
        animationJob.scale = scale;
        m_AnimationScriptPlayable.SetJobData(animationJob);
    }


    void OnDisable()
    {
        if (sceneTransform == null)
            return;


        m_Graph.Destroy();
    }
}
```

See Also: AnimatorJobExtensions.BindSceneTransform, PropertySceneHandle, PropertyStreamHandle, and TransformStreamHandle.

Public Methods

GetLocalPosition	Gets the position of the transform relative to the parent.
GetLocalRotation	Gets the rotation of the transform relative to the parent.
GetLocalScale	Gets the scale of the transform relative to the parent.
GetPosition	Gets the position of the transform in world space.
GetRotation	Gets the rotation of the transform in world space.
IsValid	Returns whether this is a valid handle.
SetLocalPosition	Sets the position of the transform relative to the parent.
SetLocalRotation	Sets the rotation of the transform relative to the parent.
SetLocalScale	Sets the scale of the transform relative to the parent.
SetPosition	Sets the position of the transform in world space.
SetRotation	Sets the rotation of the transform in world space.

Description

Position, rotation and scale of an object in the AnimationStream.

```
using UnityEngine;
using UnityEngine.Playables;
using UnityEngine.Animations;


using UnityEngine.Experimental.Animations;


public struct TransformStreamHandleJob : IAnimationJob
{
    public TransformStreamHandle handle;
    public Vector3 position;
    public Vector3 rotation;
    public Vector3 scale;


    public void ProcessRootMotion(AnimationStream stream)
    {
        // Set the new local position.
        handle.SetLocalPosition(stream, position);


        // Set the new local rotation (converted from euler).
        handle.SetLocalRotation(stream, Quaternion.Euler(rotation));


        // Set the new local scale.
        handle.SetLocalScale(stream, scale);
    }


    public void ProcessAnimation(AnimationStream stream)
    {
    }
}


[RequireComponent(typeof(Animator))]
public class TransformStreamHandleExample : MonoBehaviour
```

```
{
    public Vector3 position;
    public Vector3 rotation;
    public Vector3 scale = Vector3.one;


    PlayableGraph m_Graph;
    AnimationScriptPlayable m_AnimationScriptPlayable;


    void Start()
    {
        var animator = GetComponent<Animator>();


        m_Graph = PlayableGraph.Create("TransformStreamHandleExample");
        var output = AnimationPlayableOutput.Create(m_Graph, "output", animator)


        var animationJob = new TransformStreamHandleJob();
        animationJob.handle = animator.BindStreamTransform(gameObject.transform)
        m_AnimationScriptPlayable = AnimationScriptPlayable.Create(m_Graph, anim


        output.SetSourcePlayable(m_AnimationScriptPlayable);
        m_Graph.Play();
    }


    void Update()
    {
        var animationJob = m_AnimationScriptPlayable.GetJobData<TransformStreamH
        animationJob.position = position;
        animationJob.rotation = rotation;
        animationJob.scale = scale;
        m_AnimationScriptPlayable.SetJobData(animationJob);
    }


    void OnDisable()
    {
        m_Graph.Destroy();
    }
}
```

See Also: AnimatorJobExtensions.BindStreamTransform, PropertyStreamHandle, PropertySceneHandle, and TransformSceneHandle.

Public Methods

GetLocalPosition	Gets the position of the transform relative to the parent.
GetLocalRotation	Gets the rotation of the transform relative to the parent.
GetLocalScale	Gets the scale of the transform relative to the parent.
GetPosition	Gets the position of the transform in world space.
GetRotation	Gets the rotation of the transform in world space.
IsResolved	Returns whether this handle is resolved.
IsValid	Returns whether this is a valid handle.
Resolve	Bind this handle with an animated values from the AnimationStream.
SetLocalPosition	Sets the position of the transform relative to the parent.
SetLocalRotation	Sets the rotation of the transform relative to the parent.
SetLocalScale	Sets the scale of the transform relative to the parent.
SetPosition	Sets the position of the transform in world space.
SetRotation	Sets the rotation of the transform in world space.

Description

The interface defining an animation job to use with an IAnimationJobPlayable.

See Also: AnimationScriptPlayable.Create.

Public Methods

ProcessAnimation	Defines what to do when processing the animation.
ProcessRootMotion	Defines what to do when processing the root motion.

Implements interfaces:IPlayable

Description

The interface defining an animation playable that uses IAnimationJob.

See Also: AnimationScriptPlayable.

Public Methods

GetJobData	Gets the job data contained in the playable.
SetJobData	Sets a new job data in the playable.

Description

Provides access to the audio samples generated by Unity objects such as VideoPlayer.

This class is meant to be usable outside the main thread and the implementation assumes a single consumer at a time invoking AudioSampleProvider.ConsumeSampleFrames. Audio data can be accessed both from C# and from C++, depending on what has to be done with this data.

The following is an example of how to access samples from C# using an accessor obtained from the VideoPlayer:

```
using UnityEngine;
using Unity.Collections;
using UnityEngine.Experimental.Video;
using UnityEngine.Experimental.Audio;
using UnityEngine.Video;


public class ManagedAudioOutput : MonoBehaviour
{
    AudioSampleProvider provider;


    void Start()
    {
        VideoPlayer vp = GetComponent<VideoPlayer>();
        vp.audioOutputMode = VideoAudioOutputMode.APIOnly;
        vp.prepareCompleted += Prepared;
        vp.Prepare();
    }


    void Prepared(VideoPlayer vp)
    {
        provider = vp.GetAudioSampleProvider(0);
        provider.sampleFramesAvailable += SampleFramesAvailable;
        provider.enableSampleFramesAvailableEvents = true;
        provider.freeSampleFrameCountLowThreshold = provider.maxSampleFrameCount
        vp.Play();
    }


    void SampleFramesAvailable(AudioSampleProvider provider, uint sampleFrameCou
    {
```

```
        using (NativeArray<float> buffer = new NativeArray<float>(
                        (int)sampleFrameCount * provider.channelCount, Allocator.T
        {
            var sfCount = provider.ConsumeSampleFrames(buffer);
            Debug.LogFormat("SetupSoftwareAudioOut.Available got {0} sample fram
            // Do something with the samples here...
        }
    }
}
```

The following is an example of how to access samples from C++. The setup has to be done in C# and then Unity's core and the native plug-in can call into each other without having to go through managed code.

```
using System;
using System.Runtime.InteropServices;
using UnityEngine;
using UnityEngine.Experimental.Video;
using UnityEngine.Experimental.Audio;
using UnityEngine.Video;


public class NativeAudioOutput : MonoBehaviour
{
    AudioSampleProvider provider;
    AudioSampleProvider.SampleFramesEventNativeFunction sampleFramesAvailableNat
        SampleFramesAvailable;


    void Start()
    {
        VideoPlayer vp = GetComponent<VideoPlayer>();
        vp.audioOutputMode = VideoAudioOutputMode.APIOnly;
        vp.prepareCompleted += Prepared;
        vp.Prepare();
    }


    void Prepared(VideoPlayer vp)
    {
        provider = vp.GetAudioSampleProvider(0);
        provider.freeSampleFrameCountLowThreshold = provider.maxSampleFrameCount


        SetConsumeSampleFramesFunction(
```

```
            AudioSampleProvider.consumeSampleFramesNativeFunction, provider.id,
            provider.channelCount, provider.sampleRate);
        provider.SetSampleFramesAvailableNativeHandler(
            sampleFramesAvailableNativeHandler, (IntPtr)0);


        vp.Play();
    }


    private const string pluginName =
#if UNITY_IPHONE
        "__Internal"
#else
        "NativeAudioOutputPlugin"
#endif
    ;


    [DllImport(pluginName, ExactSpelling = true, CallingConvention = CallingConv
    private static extern void SetConsumeSampleFramesFunction(
        AudioSampleProvider.ConsumeSampleFramesNativeFunction cb, uint id, ushor


    [AOT.MonoPInvokeCallback(typeof(AudioSampleProvider.SampleFramesEventNativeF
    [DllImport(pluginName, ExactSpelling = true, CallingConvention = CallingConv
    private static extern void SampleFramesAvailable(IntPtr userData, uint id, u
}
```

And here is the accompanying C++ plug-in:

```
#include <algorithm>
#include <stdint.h>


typedef uint32_t(__cdecl *ConsumeSampleFramesFunction)(
    uint32_t providerId, float* interleavedSampleFrames, uint32_t sampleFrameCou


ConsumeSampleFramesFunction ConsumeSampleFrames = NULL;
uint32_t providerId = -1;
float* buffer = NULL;
uint32_t bufferSampleFrameCount = 0;
uint32_t availableSampleFrameCount = 0;
```

```
extern "C" __cdecl void SetConsumeSampleFramesFunction(
    ConsumeSampleFramesFunction function, uint32_t id, uint16_t channelCount, ui
{
    ConsumeSampleFrames = function;
    providerId = id;
    delete[] buffer;
    buffer = new float[channelCount * sampleRate]; // 1s worth of sample frames.
    bufferSampleFrameCount = sampleRate;
}


extern "C" __cdecl void SampleFramesAvailable(void* userData, uint32_t id, uint3
{
    if (ConsumeSampleFrames == NULL)
        return;


    // We consume the sample frames from the handler that tells us that there ar
    // But we could also invoke this regularly from another thread, for example
    // samples to an audio device.
    const uint32_t consumedSampleFrameCount = ConsumeSampleFrames(
            providerId, buffer, std::min(bufferSampleFrameCount, sampleFrameCoun
    // Do something with the samples here...
}
```

Static Properties

consumeSampleFramesNativeFunction	Pointer to the native function that provides access to audio sample frames.

Properties

availableSampleFrameCount	Number of sample frames available for consuming with AudioSampleProvider.ConsumeSampleFrames.
channelCount	The number of audio channels per sample frame.
enableSampleFramesAvailableEvents	Enables the AudioSampleProvider.sampleFramesAvailable events.
enableSilencePadding	If true, buffers produced by ConsumeSampleFrames will get padded when silence if there are less available than asked for. Otherwise, the extra sample frames in the buffer will be left unchanged.

freeSampleFrameCount	Number of sample frames that can still be written to by the sample producer before overflowing.
freeSampleFrameCountLowThreshold	Then the free sample count falls below this threshold, the AudioSampleProvider.sampleFramesAvailable event and associated native is emitted.
id	Unique identifier for this instance.
maxSampleFrameCount	The maximum number of sample frames that can be accumulated inside the internal buffer before an overflow event is emitted.
owner	Object where this provider came from.
sampleRate	The expected playback rate for the sample frames produced by this class.
trackIndex	Index of the track in the object that created this provider.
valid	True if the object is valid.

Public Methods

ClearSampleFramesAvailableNativeHandler	Clear the native handler set with AudioSampleProvider.SetSampleFramesAvailableNativeHandler.
ClearSampleFramesOverflowNativeHandler	Clear the native handler set with AudioSampleProvider.SetSampleFramesOverflowNativeHandler.
ConsumeSampleFrames	Consume sample frames from the internal buffer.
Dispose	Release internal resources. Inherited from IDisposable.
SetSampleFramesAvailableNativeHandler	Set the native event handler for events emitted when the number of available sample frames crosses the threshold.
SetSampleFramesOverflowNativeHandler	Set the native event handler for events emitted when the internal sample frame buffer overflows.

Events

sampleFramesAvailable	Invoked when the number of available sample frames goes beyond the threshold set with AudioSampleProvider.freeSampleFrameCountLowThreshold.
sampleFramesOverflow	Invoked when the number of available sample frames goes beyond the maximum that fits in the internal buffer.

Delegates

ConsumeSampleFramesNativeFunction	Type that represents the native function pointer for consuming sample frames.
SampleFramesEventNativeFunction	Type that represents the native function pointer for handling sample frame events.
SampleFramesHandler	Delegate for sample frame events.

Description

A helper structure used to initialize a LightDataGI structure as a directional light.

Properties

color	The direct light color.
direction	The direction of the light.
indirectColor	The indirect light color.
instanceID	The light's instanceID.
mode	The lightmode.
penumbraWidthRadian	The penumbra width for soft shadows in radians.
shadow	True if the light casts shadows, otherwise False.

Description

The interop structure to pass light information to the light baking backends. There are helper structures for Directional, Point, Spot and Rectangle lights to correctly initialize this structure.

Properties

color	The color of the light.
coneAngle	The cone angle for spot lights.
falloff	The falloff model to use for baking point and spot lights.
indirectColor	The indirect color of the light.
innerConeAngle	The inner cone angle for spot lights.
instanceID	The light's instanceID.
mode	The lightmap mode for the light.
orientation	The orientation of the light.
position	The position of the light.
range	The range of the light. Unused for directional lights.
shadow	Set to 1 for shadow casting lights, 0 otherwise.
shape0	The light's sphere radius for point and spot lights, or the width for rectangle lights.
shape1	The height for rectangle lights.
type	The type of the light.

Public Methods

Init	Initialize the struct with the parameters from the given light type.
InitNoBake	Initialize a light so that the baking backends ignore it.

Description

Utility class for converting Unity Lights to light types recognized by the baking backends.

Static Methods

Extract	Extracts informations from Lights.
ExtractIndirect	Extracts the indirect color from a light.
ExtractInnerCone	Extracts the inner cone angle of spot lights.

Description

Interface to the light baking backends.

Static Methods

GetDelegate	Get the currently set conversion delegate.
ResetDelegate	Resets the light conversion delegate to Unity's default conversion function.
SetDelegate	Set a delegate that converts a list of lights to a list of LightDataGI structures that are passed to the baking backends. Must be reset by calling ResetDelegate again.

Delegates

RequestLightsDelegate	Delegate called when converting lights into a form that the baking backends understand.

Description

Contains normalized linear color values for red, green, blue in the range of 0 to 1, and an additional intensity value.

Properties

blue	The blue color value in the range of 0.0 to 1.0.
green	The green color value in the range of 0.0 to 1.0.
intensity	The intensity value used to scale the red, green and blue values.
red	The red color value in the range of 0.0 to 1.0.

Static Methods

Black	Returns a black color.
Convert	Converts a Light's color value to a normalized linear color value, automatically handling gamma conversion if necessary.

Description

A helper structure used to initialize a LightDataGI structure as a point light.

Properties

color	The direct light color.
falloff	The falloff model to use for baking the point light.
indirectColor	The indirect light color.
instanceID	The light's instanceID.
mode	The lightmode.
position	The light's position.
range	The light's range.
shadow	True if the light casts shadows, otherwise False.
sphereRadius	The light's sphere radius, influencing soft shadows.

Description

A helper structure used to initialize a LightDataGI structure as a rectangle light.

Properties

color	The direct light color.
height	The height of the rectangle light.
indirectColor	The indirect light color.
instanceID	The light's instanceID.
mode	The lightmode.
orientation	The light's orientation.
position	The light's position.
range	The light's range.
shadow	True if the light casts shadows, otherwise False.
width	The width of the rectangle light.

Description

A helper structure used to initialize a LightDataGI structure as a spot light.

Properties

color	The direct light color.
coneAngle	The outer angle for the spot light.
falloff	The falloff model to use for baking the spot light.
indirectColor	The indirect light color.
innerConeAngle	The inner angle for the spot light.
instanceID	The light's instanceID.
mode	The lightmode.
orientation	The light's orientation.
position	The light's position.
range	The light's range.
shadow	True if the light casts shadows, otherwise False.
sphereRadius	The light's sphere radius, influencing soft shadows.

Description

Available falloff models for baking.

Properties

InverseSquared	Inverse squared distance falloff model.
InverseSquaredNoRangeAttenuation	Inverse squared distance falloff model (without smooth range attenuation).
Linear	Linear falloff model.
Legacy	Quadratic falloff model.
Undefined	Falloff model is undefined.

Description

The lightmode. A light can be realtime, mixed, baked or unknown. Unknown lights will be ignored by the baking backends.

Properties

Realtime	The light is realtime. No contribution will be baked in lightmaps or light probes.
Mixed	The light is mixed. Mixed lights are interpreted based on the global light mode setting in the lighting window.
Baked	The light is fully baked and has no realtime component.
Unknown	The light should be ignored by the baking backends.

Description

The light type.

Properties

Directional	An infinite directional light.
Point	A point light emitting light in all directions.
Spot	A spot light emitting light in a direction with a cone shaped opening angle.
Rectangle	A light shaped like a rectangle emitting light into the hemisphere that it is facing.
Disc	A light shaped like a disc emitting light into the hemisphere that it is facing.

Description

The class representing the player loop in Unity.

This class contains functions for interacting with the player loop in the core of Unity. You can use this class to get the update order of all native systems and set a custom order with new script entry points inserted.

Static Methods

GetDefaultPlayerLoop	Returns the default update order of all engine systems in Unity.
SetPlayerLoop	Set a new custom update order of all engine systems in Unity.

Description

The representation of a single system being updated by the player loop in Unity.

This struct represents a single system in the player loop. A system can be one of Unity's built-in native systems, or you can new custom systems to add to the player loop using C#. Native systems can only be retrieved by calling PlayerLoop.GetDefaultPlayerLoop, and the parameters of them should not be modified.

Properties

loopConditionFunction	The loop condition for a native engine system. To get a valid value for this, you must copy it from one of the PlayerLoopSystems returned by PlayerLoop.GetDefaultPlayerLoop.
subSystemList	A list of sub systems which run as part of this item in the player loop.
type	This property is used to identify which native system this belongs to, or to get the name of the managed system to show in the profiler.
updateDelegate	A managed delegate. You can set this to create a new C# entrypoint in the player loop.
updateFunction	A native engine system. To get a valid value for this, you must copy it from one of the PlayerLoopSystems returned by PlayerLoop.GetDefaultPlayerLoop.

Implements interfaces:IPlayable

Description

An implementation of IPlayable that produces a Camera texture.

NOTE: You can use PlayableExtensions methods with CameraPlayable objects.

Static Methods

Create	Creates a CameraPlayable in the PlayableGraph.

Implements interfaces:IPlayable

Description

An implementation of IPlayable that allows application of a Material shader to one or many texture inputs to produce a texture output.

NOTE: You can use PlayableExtensions methods with MaterialEffectPlayable objects.

Static Methods

Create	Creates a MaterialEffectPlayable in the PlayableGraph.

Implements interfaces:IPlayable

Description

An implementation of IPlayable that allows mixing two textures.

NOTE: You can use PlayableExtensions methods with TextureMixerPlayable objects.

Static Methods

Create	Creates a TextureMixerPlayable in the PlayableGraph.

Description

A PlayableBinding that contains information representing a TexturePlayableOutput.

Static Methods

Create	Creates a PlayableBinding that contains information representing a TexturePlayableOutput.

Implements interfaces:IPlayableOutput

Description

An IPlayableOutput implementation that will be used to manipulate textures.

NOTE: You can use PlayableOutputExtensions methods with TexturePlayableOutput objects.

Static Properties

Null	Returns an invalid TexturePlayableOutput.

Description

Update phase in the native player loop.

This is the C# representation of an update phase in the native player loop. It can only be used to identify the update phase in native.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Update phase in the native player loop.

This is the C# representation of an update phase in the native player loop. It can only be used to identify the update phase in native.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Update phase in the native player loop.

This is the C# representation of an update phase in the native player loop. It can only be used to identify the update phase in native.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Update phase in the native player loop.

This is the C# representation of an update phase in the native player loop. It can only be used to identify the update phase in native.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Update phase in the native player loop.

This is the C# representation of an update phase in the native player loop. It can only be used to identify the update phase in native.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Update phase in the native player loop.

This is the C# representation of an update phase in the native player loop. It can only be used to identify the update phase in native.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Update phase in the native player loop.

This is the C# representation of an update phase in the native player loop. It can only be used to identify the update phase in native.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Native engine system updated by the native player loop.

Description

Values for the blend state.

See Also: RenderStateBlock, ShaderLab: Blending.

Static Properties

Default	Default values for the blend state.

Properties

alphaToMask	Turns on alpha-to-coverage.
blendState0	Blend state for render target 0.
blendState1	Blend state for render target 1.
blendState2	Blend state for render target 2.
blendState3	Blend state for render target 3.
blendState4	Blend state for render target 4.
blendState5	Blend state for render target 5.
blendState6	Blend state for render target 6.
blendState7	Blend state for render target 7.
separateMRTBlendStates	Determines whether each render target uses a separate blend state.

Constructors

BlendState	Creates a new blend state with the specified values.

Description

Camera related properties in CullingParameters.

Typical use case is to compute culling parameters from Camera data, see CullResults.GetCullingParameters.

Public Methods

GetCameraCullingPlane	Get a camera culling plane.
GetShadowCullingPlane	Get a shadow culling plane.
SetCameraCullingPlane	Set a camera culling plane.
SetShadowCullingPlane	Set a shadow culling plane.

Description

Core Camera related properties in CullingParameters.

Typical use case is to compute culling parameters from Camera data, see CullResults.GetCullingParameters.

Description

Culling results (visible objects, lights, reflection probes).

In a scriptable render loop, the rendering process is typically doing culling for each camera (CullResults.Cull), followed by rendering subsets of visible objects (RenderLoop.DrawRenderers) and processing visible lights (visibleLights, visibleReflectionProbes).

CullResults also provides several functions to aid shadow rendering (ComputeDirectionalShadowMatricesAndCullingPrimitives, ComputeSpotShadowMatricesAndCullingPrimitives, ComputePointShadowMatricesAndCullingPrimitives).

Properties

visibleLights	Array of visible lights.
visibleOffscreenVertexLights	Off screen lights that still effect visible Scene vertices.
visibleReflectionProbes	Array of visible reflection probes.
visibleRenderers	Visible renderers.

Public Methods

ComputeDirectionalShadowMatricesAndCullingPrimitives	Calculates the view and projection matrices and shadow split data for a directional light.
ComputePointShadowMatricesAndCullingPrimitives	Calculates the view and projection matrices and shadow split data for a point light.
ComputeSpotShadowMatricesAndCullingPrimitives	Calculates the view and projection matrices and shadow split data for a spot light.
FillLightIndices	Fills a compute buffer with per-object light indices.
GetLightIndexMap	If a RenderPipeline sorts or otherwise modifies the VisibleLight list, an index remap will be necessary to properly make use of per-object light lists.
GetLightIndicesCount	Gets the number of per-object light indices.
GetShadowCasterBounds	Returns the bounding box that encapsulates the visible shadow casters. Can be used to, for instance, dynamically adjust cascade ranges.

SetLightIndexMap	If a RenderPipeline sorts or otherwise modifies the VisibleLight list, an index remap will be necessary to properly make use of per-object light lists. If an element of the array is set to -1, the light corresponding to that element will be disabled.

Static Methods

Cull	Perform culling for a Camera.
GetCullingParameters	Get culling parameters for a camera.

Description

Values for the depth state.

See Also: RenderStateBlock, ShaderLab: Culling & Depth Testing.

Static Properties

Default	Default values for the depth state.

Properties

compareFunction	How should depth testing be performed.
writeEnabled	Controls whether pixels from this object are written to the depth buffer.

Constructors

DepthState	Creates a new depth state with the given values.

Description

Settings for ScriptableRenderContext.DrawRenderers.

DrawRendererSettings describes how to sort visible objects are sorted (sorting) and which shader passes to use (shaderPassName).

See Also: ScriptableRenderContext.DrawRenderers, CullResults.Cull, FilterRendererSettings.

Static Properties

maxShaderPasses	The maxiumum number of passes that can be rendered in 1 DrawRenderers call.

Properties

flags	Other flags controlling object rendering.
rendererConfiguration	What kind of per-object data to setup during rendering.
sorting	How to sort objects during rendering.

Constructors

DrawRendererSettings	Create a draw settings struct.

Public Methods

SetOverrideMaterial	Set the Material to use for all drawers that would render in this group.
SetShaderPassName	Set the shader passes that this draw call can render.

Description

This struct describes the methods to sort objects during rendering.

See Also: DrawRendererSettings, RenderLoop.DrawRenderers.

Properties

cameraCustomSortAxis	Used to calculate distance to objects, by comparing the positions of objects to this axis.
cameraPosition	Used to calculate the distance to objects.
flags	What kind of sorting to do while rendering.
sortMode	Type of sorting to use while rendering.
worldToCameraMatrix	Used to calculate the distance to objects.

Description

Settings for ScriptableRenderContext.DrawShadows.

This structure describes which shadow light to render (lightIndex) with what split settings (splitData).

See Also: ShadowSplitData.

Properties

cullResults	Culling results to use.
lightIndex	The index of the shadow-casting light to be rendered.
splitData	The split data.

Constructors

DrawShadowsSettings	Create a shadow settings object.

Description

Filter settings for ScriptableRenderContext.DrawRenderers.

FilterRendererSettings describes how to filter the given set of visible objects to render.

See Also: ScriptableRenderContext.DrawRenderers.

Properties

excludeMotionVectorObjects	Set to true to exclude objects that are currently in motion from rendering. The default value is false.
layerMask	Only render objects in the given layer mask.
renderingLayerMask	The rendering layer mask to use when filtering available renderers for drawing.
renderQueueRange	Render objects whose material render queue in inside this range.

Constructors

FilterRendererSettings

Description

Describes a subset of objects to be rendered.

See Also: ScriptableRenderContext.DrawRenderers.

.

Description

LODGroup culling parameters.

See Also: CullingParameters.

Properties

cameraPixelHeight	Rendering view height in pixels.
cameraPosition	Camera position.
fieldOfView	Camera's field of view.
isOrthographic	Indicates whether camera is orthographic.
orthoSize	Orhographic camera size.

Description

Values for the raster state.

See Also: ShaderLab: Culling & Depth Testing.

Static Properties

Default	Default values for the raster state.

Properties

cullingMode	Controls which sides of polygons should be culled (not drawn).
depthClip	Enable clipping based on depth.
offsetFactor	Scales the maximum Z slope.
offsetUnits	Scales the minimum resolvable depth buffer value.

Constructors

RasterState	Creates a new raster state with the given values.

Description

Object encapsulating the duration of a single renderpass that contains one or more subpasses.

The RenderPass object provides a new way to switch rendertargets in the context of a Scriptable Rendering Pipeline. As opposed to the SetRenderTargets function, the RenderPass object specifies a clear beginning and an end for the rendering, alongside explicit load/store actions on the rendering surfaces.

The RenderPass object also allows running multiple subpasses within the same renderpass, where the pixel shaders have a read access to the current pixel value within the renderpass. This allows for efficient implementation of various rendering methods on tile-based GPUs, such as deferred rendering.

RenderPasses are natively implemented on Metal (iOS) and Vulkan, but the API is fully functional on all rendering backends via emulation (using legacy SetRenderTargets calls and reading the current pixel values via texel fetches).

A quick example on how to use the RenderPass API within the Scriptable Render Pipeline to implement deferred rendering:

The RenderPass mechanism has the following limitations: - All attachments must have the same resolution and MSAA sample count - The rendering results of previous subpasses are only available within the same screen-space pixel coordinate via the UNITY_READ_FRAMEBUFFER_INPUT(x) macro in the shader; the attachments cannot be bound as textures or otherwise accessed until the renderpass has ended - iOS Metal does not allow reading from the Z-Buffer, so an additional render target is needed to work around that - The maximum amount of attachments allowed per RenderPass is currently 8 + depth, but note that various GPUs may have stricter limits.

```
using UnityEngine;
using UnityEngine.Experimental.Rendering;


public class DeferredRenderer
{
    private RenderPassAttachment m_Albedo;
    private RenderPassAttachment m_SpecRough;
    private RenderPassAttachment m_Normal;
    private RenderPassAttachment m_Emission;
    private RenderPassAttachment m_Depth;


    public DeferredRenderer()
    {
        // Create the attachment objects. If these attachments are not specifica
        // these are treated as temporary surfaces that are discarded at the end
```

```
    // NOTE: DO NOT ALLOCATE NEW RENDERPASSATTACHMENTS EVERY FRAME!
    // These objects only get garbage collected when a Scene is unloaded, so
    m_Albedo = new RenderPassAttachment(RenderTextureFormat.ARGB32);
    m_SpecRough = new RenderPassAttachment(RenderTextureFormat.ARGB32);
    m_Normal = new RenderPassAttachment(RenderTextureFormat.ARGB2101010);
    m_Emission = new RenderPassAttachment(RenderTextureFormat.ARGBHalf);
    m_Depth = new RenderPassAttachment(RenderTextureFormat.Depth);


    // At the beginning of the render pass, clear the emission buffer to all
    m_Emission.Clear(new Color(0.0f, 0.0f, 0.0f, 0.0f), 1.0f, 0);
    m_Depth.Clear(new Color(), 1.0f, 0);
}


public void ExecuteRenderLoop(Camera camera, CullResults cullResults, Script
{
    // Bind the albedo surface to the current camera target, so the final pa
    // The second argument specifies whether the existing contents of the su
    // in our case we do not need that because we'll be clearing the attachm
    // bandwidth on tiled GPUs.
    // The third argument specifies whether the rendering results need to be
    // the renderpass. We need this as we'll be generating the final image t
    // We could do this in the constructor already, but the camera target ma
    m_Albedo.BindSurface(BuiltinRenderTextureType.CameraTarget, false, true)


    // All other attachments are transient surfaces that are not stored anyw
    // those surfaces do not even have a memory allocated for the pixel valu


    // Start the renderpass using the given scriptable rendercontext, resolu
    using (RenderPass rp = new RenderPass(scriptableRenderContext, camera.pi
    {
        // Start the first subpass, GBuffer creation: render to albedo, spec
        using (new RenderPass.SubPass(rp, new[] { m_Albedo, m_SpecRough, m_N
        {
            // Render the deferred G-Buffer
            RenderGBuffer(cullResults, camera, scriptableRenderContext);
        }
        // Second subpass, lighting: Render to the emission buffer, read fro
        // The last parameter indicates whether the depth buffer can be boun
        // Note that some renderers (notably iOS Metal) won't allow reading
        // so those renderers should write the Z into an additional FP32 ren
        using (new RenderPass.SubPass(rp, new[] { m_Emission }, new[] { m_Al
        {
```

```
                PushGlobalShadowParams(scriptableRenderContext);


                RenderLighting(camera, cullResults, scriptableRenderContext);


                scriptableRenderContext.DrawSkybox(camera);
            }
            // Third subpass, tonemapping: Render to albedo (which is bound to t
            using (new RenderPass.SubPass(rp, new[] { m_Albedo }, new[] { m_Emis
            {
                // present frame buffer.
                FinalPass(scriptableRenderContext);
            }
        }
    }
}
```

Properties

colorAttachments	Read only: array of RenderPassAttachment objects currently bound into this RenderPass.
context	Read only: The ScriptableRenderContext object this RenderPass was created for.
depthAttachment	Read only: The depth/stencil attachment used in this RenderPass, or null if none.
height	Read only: The height of the RenderPass surfaces in pixels.
sampleCount	Read only: MSAA sample count for this RenderPass.
width	Read only: The width of the RenderPass surfaces in pixels.

Constructors

RenderPass	Create a RenderPass and start it within the ScriptableRenderContext.

Public Methods

Dispose	End the RenderPass.

Description

This class encapsulates a single subpass within a RenderPass. RenderPasses can never be standalone, they must always contain at least one SubPass. See Also: RenderPass.

Constructors

RenderPass.SubPass	Create a subpass and start it.

Public Methods

Dispose	End the subpass.

Description

A declaration of a single color or depth rendering surface to be attached into a RenderPass.

A RenderPassAttachment object identifies a single color or depth rendering surface that can be used with a RenderPass. Note that the RenderPassAttachment object derives from UnityEngine.Object so they do not get garbage collected like normal C# objects. Instead, they are only GC'd when unloading a Scene or when Resources.UnloadUnusedAssets() is called. Therefore do not create these objects each frame: Instead, create these objects in the constructor of your rendering class, and reuse those objects each frame.

Properties

clearColor	The currently assigned clear color for this attachment. Default is black.
clearDepth	Currently assigned depth clear value for this attachment. Default value is 1.0.
clearStencil	Currently assigned stencil clear value for this attachment. Default is 0.
format	The RenderTextureFormat of this attachment.
loadAction	The load action to be used on this attachment when the RenderPass starts.
storeAction	The store action to use with this attachment when the RenderPass ends. Only used when either BindSurface or BindResolveSurface has been called.

Constructors

RenderPassAttachment	Create a RenderPassAttachment to be used with RenderPass.

Public Methods

BindResolveSurface	When the renderpass that uses this attachment ends, resolve the MSAA surface into the given target.
BindSurface	Binds this RenderPassAttachment to the given target surface.
Clear	When the RenderPass starts, clear this attachment into the color or depth/stencil values given (depending on the format of this attachment). Changes loadAction to RenderBufferLoadAction.Clear.

Inherited Members

Properties

hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.

Public Methods

GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.

Static Methods

Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Implements interfaces:IRenderPipeline

Description

Defines a series of commands and settings that describes how Unity renders a frame.

Properties

disposed	When the IRenderPipeline is invalid or destroyed this returns true.

Public Methods

Dispose	Dispose the Renderpipeline destroying all internal state.
Render	Defines custom rendering for this RenderPipeline.

Static Methods

BeginCameraRendering	Call the delegate used during SRP rendering before a single camera starts rendering.
BeginFrameRendering	Call the delegate used during SRP rendering before a render begins.

Events

beginCameraRendering	Call that should be issued by an SRP when the SRP begins to render a Camera so that other systems can inject per camera render logic.
beginFrameRendering	Call that should be issued by an SRP when the SRP begins to render so that other systems can inject 'pre render' logic.

Implements interfaces:IRenderPipelineAsset

Description

An asset that produces a specific IRenderPipeline.

Default implementation of IRenderPipelineAsset. This manages the lifecylces of inherited types, as well as ensures that created IRenderPipeline's are invalidated when the asset is changed.

See Also: IRenderPipelineAsset.

Public Methods

CreatePipeline	Create a IRenderPipeline specific to this asset.
DestroyCreatedInstances	Destroys all cached data and created IRenderLoop's.
GetAutodeskInteractiveMaskedShader	Retrieves the default Autodesk Interactive masked Shader for this pipeline.
GetAutodeskInteractiveShader	Retrieves the default Autodesk Interactive Shader for this pipeline.
GetAutodeskInteractiveTransparentShader	Retrieves the default Autodesk Interactive transparent Shader for this pipeline.
GetDefault2DMaterial	Return the default 2D Material for this pipeline.
GetDefaultLineMaterial	Return the default Line Material for this pipeline.
GetDefaultMaterial	Return the default Material for this pipeline.
GetDefaultParticleMaterial	Return the default particle Material for this pipeline.
GetDefaultShader	Return the default Shader for this pipeline.
GetDefaultTerrainMaterial	Return the default Terrain Material for this pipeline.
GetDefaultUIETC1SupportedMaterial	Return the default UI ETC1 Material for this pipeline.
GetDefaultUIMaterial	Return the default UI Material for this pipeline.
GetDefaultUIOverdrawMaterial	Return the default UI overdraw Material for this pipeline.
GetRenderingLayerMaskNames	Returns the list of names used to display Rendering Layer Mask UI for this pipeline.
GetTerrainBrushPassIndex	The render index for the terrain brush in the editor.

Protected Methods

CreatedInstances	Returns the list of current IRenderPipeline's created by the asset.
InternalCreatePipeline	Create a IRenderPipeline specific to this asset.
OnDisable	Default implementation of OnDisable for RenderPipelineAsset. See ScriptableObject.OnDisable
OnValidate	Default implementation of OnValidate for RenderPipelineAsset. See MonoBehaviour.OnValidate

Inherited Members

Properties

hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.

Public Methods

GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.

Static Methods

Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.
CreateInstance	Creates an instance of a scriptable object.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.

operator ==	Compares two object references to see if they refer to the same object.

Messages

Awake	This function is called when the ScriptableObject script is started.
OnDestroy	This function is called when the scriptable object will be destroyed.
OnEnable	This function is called when the object is loaded.

Description

Render Pipeline manager.

Static Properties

currentPipeline	Returns the instance of the currently used Render Pipeline.

Description

Describes a material render queue range.

See Also: ScriptableRenderContext.DrawRenderers, FilterRendererSettings.renderQueueRange.

Static Properties

all	A range that includes all objects.
opaque	A range that includes only opaque objects.
transparent	A range that includes only transparent objects.

Properties

max	Inclusive upper bound for the range.
min	Inclusive lower bound for the range.

Description

A set of values used to override the render state. Note that it is not enough to set e.g. *blendState*, but that *mask* must also include RenderStateMask.Blend for the override to occur.

See Also: ScriptableRenderContext.DrawRenderers, RenderStateMask, RenderStateMapping.

Properties

blendState	Specifies the new blend state.
depthState	Specifies the new depth state.
mask	Specifies which parts of the render state that is overriden.
rasterState	Specifies the new raster state.
stencilReference	The value to be compared against and/or the value to be written to the buffer based on the stencil state.
stencilState	Specifies the new stencil state.

Constructors

RenderStateBlock	Creates a new render state block with the specified mask.

Description

Maps a RenderType to a specific render state override.

See Also: ScriptableRenderContext.DrawRenderers, RenderStateBlock.

Properties

renderType	Specifices the RenderType to override the render state for.
stateBlock	Specifies the values to override the render state with.

Constructors

RenderStateMapping	Creates a new render state mapping with the specified values.

Description

Values for the blend state.

See Also: RenderStateBlock, ShaderLab: Blending.

Static Properties

Default	Default values for the blend state.

Properties

alphaBlendOperation	Operation used for blending the alpha (A) channel.
colorBlendOperation	Operation used for blending the color (RGB) channel.
destinationAlphaBlendMode	Blend factor used for the alpha (A) channel of the destination.
destinationColorBlendMode	Blend factor used for the color (RGB) channel of the destination.
sourceAlphaBlendMode	Blend factor used for the alpha (A) channel of the source.
sourceColorBlendMode	Blend factor used for the color (RGB) channel of the source.
writeMask	Specifies which color components will get written into the target framebuffer.

Constructors

RenderTargetBlendState	Creates a new blend state with the given values.

Description

Parameters controlling culling process in CullResults.

Typical use case is to compute culling parameters from Camera data, see CullResults.GetCullingParameters.

Properties

accurateOcclusionThreshold	This parameter determines query distance for occlusion culling. The accurateOcclusionThreshold controls the distance where the level of detail (LOD) changes.The default value of this parameter is -1, and any value less than 0 has the same effect. Default values result in automatic calculation of the LOD.When you use occlusion culling, the occlusion data of the world varies in level of detail. In the occlusion data, there are tiles of various sizes. Each tile contains a cells-and-portals graph. In each cell, visibility is the same. This means that any two points are visible within the cell. Portals are the openings between the cells, which determine the visibility between them.The tiles are in a k-d tree. The tree contains different sized tiles, where each tile represents a level of detail. When you query a small tile, you get accurate culling results at the price of query time.During the culling, the tile size varies with the distance from the camera. This gives finer detail closer to the camera, and coarser detail at further distance.The higher the value is, the higher the accuracy is far away form the camera. High values can have a negative impact on performance.
cameraProperties	Camera Properties used for culling.
cullingFlags	Culling Flags for the culling.
cullingMask	CullingMask used for culling.
cullingMatrix	CullingMatrix used for culling.
cullingPlaneCount	Number of culling planes to use.
cullStereoProj	The projection matrix generated for single-pass stereo culling.
cullStereoSeparation	Distance between the virtual eyes.
cullStereoView	The view matrix generated for single-pass stereo culling.
isOrthographic	Is the cull orthographic.
layerCull	Layers to cull.
lodParameters	LODParameters for culling.
position	Position for the origin of th cull.

reflectionProbeSortOptions	Reflection Probe Sort options for the cull.
sceneMask	Scene Mask to use for the cull.
shadowDistance	Shadow distance to use for the cull.

Public Methods

GetCullingPlane	Fetch the culling plane at the given index.
GetLayerCullDistance	Get the distance for the culling of a specific layer.
SetCullingPlane	Set the culling plane at a given index.
SetLayerCullDistance	Set the distance for the culling of a specific layer.

Description

Defines state and drawing commands used in a custom render pipelines.

When defining a custom RenderPipeline, a ScriptableRenderContext is used to set state and drawing commands to be submitted to the GPU.

A RenderPipeline.Render method implementation typically culls objects that don't need to be rendered for all the Cameras (see CullResults), and then makes a series of calls to ScriptableRenderContext.DrawRenderers intermixed with ScriptableRenderContext.ExecuteCommandBuffer calls. These calls set up global Shader properties, change render targets, dispatch compute shaders, and other rendering tasks. Finally, ScriptableRenderContext.Submit is called to execute the render loop.

See Also: RenderPipeline.

Public Methods

DrawRenderers	Draw subset of visible objects.
DrawShadows	Draw shadow casters for a single light.
DrawSkybox	Draw skybox.
ExecuteCommandBuffer	Execute a custom graphics command buffer.
ExecuteCommandBufferAsync	Executes a command buffer on an async compute queue with the queue selected based on the ComputeQueueType parameter passed. execution:CommandBuffer.BeginSampleCommandBuffer.CopyCounterValueCommandBuffer.CopyTextureCommandBuffer.CreateGPUF of the commands within the buffer are guaranteed to be executed on the same queue. If the target platform does not support async cor
SetupCameraProperties	Setup camera specific global shader variables.
StartMultiEye	Fine-grain control to begin stereo rendering on the scriptable render context.
StereoEndRender	Indicate completion of stereo rendering on a single frame.
StopMultiEye	Stop stereo rendering on the scriptable render context.
Submit	Submit rendering loop for execution.

Static Methods

EmitWorldGeometryForSceneView	Emit UI geometry into the Scene view for rendering.

Description

Shader pass name identifier.

See Also: DrawRendererSettings, RenderLoop.DrawRenderers.

Constructors

ShaderPassName	Create shader pass name identifier.

Description

Describes the culling information for a given shadow split (e.g. directional cascade).

Properties

cullingPlaneCount	The number of culling planes.
cullingSphere	The culling sphere. The first three components of the vector describe the sphere center, and the last component specifies the radius.

Public Methods

GetCullingPlane	Gets a culling plane.
SetCullingPlane	Sets a culling plane.

Description

Values for the stencil state.

See Also: RenderStateBlock, ShaderLab: Stencil.

Static Properties

Default	Default values for the stencil state.

Properties

compareFunction	The function used to compare the reference value to the current contents of the buffer.
compareFunctionBack	The function used to compare the reference value to the current contents of the buffer for back-facing geometry.
compareFunctionFront	The function used to compare the reference value to the current contents of the buffer for front-facing geometry.
enabled	Controls whether the stencil buffer is enabled.
failOperation	What to do with the contents of the buffer if the stencil test fails.
failOperationBack	What to do with the contents of the buffer if the stencil test fails for back-facing geometry.
failOperationFront	What to do with the contents of the buffer if the stencil test fails for front-facing geometry.
passOperation	What to do with the contents of the buffer if the stencil test (and the depth test) passes.
passOperationBack	What to do with the contents of the buffer if the stencil test (and the depth test) passes for back-facing geometry.
passOperationFront	What to do with the contents of the buffer if the stencil test (and the depth test) passes for front-facing geometry.
readMask	An 8 bit mask as an 0–255 integer, used when comparing the reference value with the contents of the buffer.
writeMask	An 8 bit mask as an 0–255 integer, used when writing to the buffer.
zFailOperation	What to do with the contents of the buffer if the stencil test passes, but the depth test fails.

zFailOperationBack	What to do with the contents of the buffer if the stencil test passes, but the depth test fails for back-facing geometry.
zFailOperationFront	What to do with the contents of the buffer if the stencil test passes, but the depth test fails for front-facing geometry.

Constructors

StencilState	Creates a new stencil state with the given values.

Description

Describes the rendering features supported by a given render pipeline.

Set the active supported rendering features when enabling a render pipeline. This will change the state of the editor UI to reflect the changes.

Static Properties

active	Get / Set a SupportedRenderingFeatures.

Properties

defaultMixedLightingMode	This is the fallback mode if the mode the user had previously selected is no longer available. See SupportedRenderingFeatures.supportedMixedLightingModes.
reflectionProbeSupportFlags	Flags for supported reflection probes.
rendererOverridesEnvironmentLighting	Determines if the renderer will override the Environment Lighting and will no longer need the built-in UI for it.
rendererOverridesFog	Determines if the renderer will override the fog settings in the Lighting Panel and will no longer need the built-in UI for it.
rendererOverridesOtherLightingSettings	Determines if the renderer will override halo and flare settings in the Lighting Panel and will no longer need the built-in UI for it.
rendererSupportsLightProbeProxyVolumes	Are light probe proxy volumes supported?
rendererSupportsMotionVectors	Are motion vectors supported?
rendererSupportsReceiveShadows	Can renderers support receiving shadows?
rendererSupportsReflectionProbes	Are reflection probes supported?
rendererSupportsRendererPriority	Determines if the renderer supports renderer priority sorting.
supportedLightmapBakeTypes	What baking types are supported. The unsupported ones will be hidden from the UI. See LightmapBakeType.
supportedLightmapsModes	Specifies what modes are supported. Has to be at least one. See LightmapsMode.
supportedMixedLightingModes	Specifies what LightmapMixedBakeMode that are supported. Please define a

SupportedRenderingFeatures.defaultMixedLightingMode in case multiple modes are supported.

Description

Same as MixedLightingMode for baking, but is used to determine what is supported by the pipeline.

Properties

None	No mode is supported.
IndirectOnly	Same as MixedLightingMode.IndirectOnly but determines if it is supported by the pipeline.
Subtractive	Same as MixedLightingMode.Subtractive but determines if it is supported by the pipeline.
Shadowmask	Determines what is supported by the rendering pipeline. This enum is similar to MixedLightingMode.

Description

Supported modes for ReflectionProbes.

Properties

None	Default reflection probe support.
Rotation	Rotated reflection probes are supported.

Description

Holds data of a visible light.

After CullResults.Cull is done, CullResults.visibleLights will contain an array of lights that are visible. The visible light structure contains packed information for most commonly used Light variables, and a VisibleLight.light reference to the Light component itself.

See Also: CullResults.visibleLights, Light.

Properties

finalColor	Light color multiplied by intensity.
flags	Light flags, see VisibleLightFlags.
light	Accessor to Light component.
lightType	Light type.
localToWorld	Light transformation matrix.
range	Light range.
screenRect	Light's influence rectangle on screen.
spotAngle	Spot light angle.

Description

Holds data of a visible reflection probe.

After CullResults.Cull is done, CullResults.visibleReflectionProbes will contain an array of reflection probes that are visible. The visible reflection probe structure contains packed information for most commonly used ReflectionProbe variables, and a VisibleReflectionProbe.probe reference to the component itself.

See Also: CullResults.visibleReflectionProbes, ReflectionProbe.

Properties

blendDistance	Probe blending distance.
bounds	Probe bounding box.
boxProjection	Should probe use box projection.
center	Probe projection center.
hdr	Shader data for probe HDR texture decoding.
importance	Probe importance.
localToWorld	Probe transformation matrix.
probe	Accessor to ReflectionProbe component.
texture	Probe texture.

Description

Defines a series of commands and settings that describes how Unity renders a frame.

Unity supports a few built-in rendering pipelines optimized for specific platforms. These pipelines are open source and available in the Github project.

You can also define your own custom RenderPipeline tailored for your specific project needs. To create a custom RenderPipeline:

1) Create a script that extends RenderPipelineAsset or IRenderPipelineAsset, this will become the manager and provider for you RenderPipeline. 2) Implement IRenderPipeline and override IRenderPipeline.Render and IRenderPipeline._disposed 3) Create a RenderPipeline Asset by using ScriptableObject.CreateAsset. 3) Use your RenderPipeline either by assigning it in the GraphicsSettings in the Editor or by using the scripting interface GraphicsSettings.renderPipeline".

Properties

disposed	When the IRenderPipeline is invalid or destroyed this returns true.

Public Methods

Render	Defines custom rendering for this RenderPipeline.

Description

An asset that produces a specific IRenderPipeline.

A RenderPipelineAsset is responsible for constructing and managing the lifecycle of a IRenderPipeline. This involves holding settings information, producing the instance used for rendering, and destroying created pipelines when the settings become invalid.

Public Methods

CreatePipeline	Create a IRenderPipeline specific to this asset.
DestroyCreatedInstances	Override this method to destroy RenderPipeline cached state.
GetTerrainBrushPassIndex	The render index for the terrain brush in the editor.

Description

Flags controlling RenderLoop.DrawRenderers.

See Also: DrawRendererSettings.flags, RenderLoop.DrawRenderers.

Properties

None	No flags are set.
EnableDynamicBatching	When set, enables dynamic batching.
EnableInstancing	When set, enables GPU instancing.

Description

Use this format usages to figure out the capabilities of specific GraphicsFormat

Each graphics card may not support all usages across formats. Use SystemInfo.IsFormatSupported to check which usages the graphics card supports.

See Also: Texture2D, texture assets.

Properties

Sample	To create and sample textures.
Linear	To sample textures with a linear filter
Render	To create and render to a rendertexture.
Blend	To blend on a rendertexture.
LoadStore	To perform resource load and store on a texture
MSAA2x	To create and render to a MSAA 2X rendertexture.
MSAA4x	To create and render to a MSAA 4X rendertexture.
MSAA8x	To create and render to a MSAA 8X rendertexture.

Description

Use this format to create either Textures or RenderTextures from scripts.

```
using UnityEngine;
using UnityEngine.Experimental.Rendering;


public class ExampleScript : MonoBehaviour
{
    TextureCreationFlags flags;
    void Start()
    {
        // Create a new texture and assign it to the material of the renderer.
        var texture = new Texture2D(128, 128, GraphicsFormat.R8G8B8A8_SRGB, flag
        GetComponent<Renderer>().material.mainTexture = texture;
    }
}
```

Each graphics card may not support all usages across formats. Use SystemInfo.IsFormatSupported to check which usages the graphics card supports.

Each "format" is represented by a single enum value. The name of a format is based on the following criteria: - For color formats, the component-format specifies the size of the R, G, B, and A components (if present). - For depth/stencil formats, the component-format specifies the size of the depth (D) and stencil (S) components (if present). - UNorm: The components are unsigned normalized values in the range [0,1]. - SNorm: The components are signed normalized values in the range [-1,1]. - UInt: The components are unsigned integer values in the range [0, 2^(n-1)]. - SInt: The components are signed integer values in the range [-2^(n-1),2^(n-1)-1]. - UFloat: The components are unsigned floating-point numbers (used by packed, shared exponent, and some compressed formats). - SFloat: The components are signed floating-point numbers. - SRGB: The R, G, and B components are unsigned normalized values that represent values using sRGB nonlinear encoding, while the A component (if one exists) is a regular unsigned normalized value. - PACKnn: The format is packed into an underlying type with nn bits.

See Also: Texture2D, texture assets.

Properties

None	The format is not specified.
R8_SRGB	A one-component, 8-bit unsigned normalized format that has a single 8-bit R component stored with sRGB nonlinear encoding.

R8G8_SRGB	A two-component, 16-bit unsigned normalized format that has an 8-bit R component stored with sRGB nonlinear encoding in byte 0, and an 8-bit G component stored with sRGB nonlinear encoding in byte 1.
R8G8B8_SRGB	A three-component, 24-bit unsigned normalized format that has an 8-bit R component stored with sRGB nonlinear encoding in byte 0, an 8-bit G component stored with sRGB nonlinear encoding in byte 1, and an 8-bit B component stored with sRGB nonlinear encoding in byte 2.
R8G8B8A8_SRGB	A four-component, 32-bit unsigned normalized format that has an 8-bit R component stored with sRGB nonlinear encoding in byte 0, an 8-bit G component stored with sRGB nonlinear encoding in byte 1, an 8-bit B component stored with sRGB nonlinear encoding in byte 2, and an 8-bit A component in byte 3.
R8_UNorm	A one-component, 8-bit unsigned normalized format that has a single 8-bit R component.
R8G8_UNorm	A two-component, 16-bit unsigned normalized format that has an 8-bit R component stored with sRGB nonlinear encoding in byte 0, and an 8-bit G component stored with sRGB nonlinear encoding in byte 1.
R8G8B8_UNorm	A three-component, 24-bit unsigned normalized format that has an 8-bit R component in byte 0, an 8-bit G component in byte 1, and an 8-bit B component in byte 2.
R8G8B8A8_UNorm	A four-component, 32-bit unsigned normalized format that has an 8-bit R component in byte 0, an 8-bit G component in byte 1, an 8-bit B component in byte 2, and an 8-bit A component in byte 3.
R8_SNorm	A one-component, 8-bit signed normalized format that has a single 8-bit R component.
R8G8_SNorm	A two-component, 16-bit signed normalized format that has an 8-bit R component stored with sRGB nonlinear encoding in byte 0, and an 8-bit G component stored with sRGB nonlinear encoding in byte 1.
R8G8B8_SNorm	A three-component, 24-bit signed normalized format that has an 8-bit R component in byte 0, an 8-bit G component in byte 1, and an 8-bit B component in byte 2.
R8G8B8A8_SNorm	A four-component, 32-bit signed normalized format that has an 8-bit R component in byte 0, an 8-bit G component in byte 1, an 8-bit B component in byte 2, and an 8-bit A component in byte 3.
R8_UInt	A one-component, 8-bit unsigned integer format that has a single 8-bit R component.
R8G8_UInt	A two-component, 16-bit unsigned integer format that has an 8-bit R component in byte 0, and an 8-bit G component in byte 1.

R8G8B8_UInt	A three-component, 24-bit unsigned integer format that has an 8-bit R component in byte 0, an 8-bit G component in byte 1, and an 8-bit B component in byte 2.
R8G8B8A8_UInt	A four-component, 32-bit unsigned integer format that has an 8-bit R component in byte 0, an 8-bit G component in byte 1, an 8-bit B component in byte 2, and an 8-bit A component in byte 3.
R8_SInt	A one-component, 8-bit signed integer format that has a single 8-bit R component.
R8G8_SInt	A two-component, 16-bit signed integer format that has an 8-bit R component in byte 0, and an 8-bit G component in byte 1.
R8G8B8_SInt	A three-component, 24-bit signed integer format that has an 8-bit R component in byte 0, an 8-bit G component in byte 1, and an 8-bit B component in byte 2.
R8G8B8A8_SInt	A four-component, 32-bit signed integer format that has an 8-bit R component in byte 0, an 8-bit G component in byte 1, an 8-bit B component in byte 2, and an 8-bit A component in byte 3.
R16_UNorm	A one-component, 16-bit unsigned normalized format that has a single 16-bit R component.
R16G16_UNorm	A two-component, 32-bit unsigned normalized format that has a 16-bit R component in bytes 0..1, and a 16-bit G component in bytes 2..3.
R16G16B16_UNorm	A three-component, 48-bit unsigned normalized format that has a 16-bit R component in bytes 0..1, a 16-bit G component in bytes 2..3, and a 16-bit B component in bytes 4..5.
R16G16B16A16_UNorm	A four-component, 64-bit unsigned normalized format that has a 16-bit R component in bytes 0..1, a 16-bit G component in bytes 2..3, a 16-bit B component in bytes 4..5, and a 16-bit A component in bytes 6..7.
R16_SNorm	A one-component, 16-bit signed normalized format that has a single 16-bit R component.
R16G16_SNorm	A two-component, 32-bit signed normalized format that has a 16-bit R component in bytes 0..1, and a 16-bit G component in bytes 2..3.
R16G16B16_SNorm	A three-component, 48-bit signed normalized format that has a 16-bit R component in bytes 0..1, a 16-bit G component in bytes 2..3, and a 16-bit B component in bytes 4..5.
R16G16B16A16_SNorm	A four-component, 64-bit signed normalized format that has a 16-bit R component in bytes 0..1, a 16-bit G component in bytes 2..3, a 16-bit B component in bytes 4..5, and a 16-bit A component in bytes 6..7.
R16_UInt	A one-component, 16-bit unsigned integer format that has a single 16-bit

	R component.
R16G16_UInt	A two-component, 32-bit unsigned integer format that has a 16-bit R component in bytes 0..1, and a 16-bit G component in bytes 2..3.
R16G16B16_UInt	A three-component, 48-bit unsigned integer format that has a 16-bit R component in bytes 0..1, a 16-bit G component in bytes 2..3, and a 16-bit B component in bytes 4..5.
R16G16B16A16_UInt	A four-component, 64-bit unsigned integer format that has a 16-bit R component in bytes 0..1, a 16-bit G component in bytes 2..3, a 16-bit B component in bytes 4..5, and a 16-bit A component in bytes 6..7.
R16_SInt	A one-component, 16-bit signed integer format that has a single 16-bit R component.
R16G16_SInt	A two-component, 32-bit signed integer format that has a 16-bit R component in bytes 0..1, and a 16-bit G component in bytes 2..3.
R16G16B16_SInt	A three-component, 48-bit signed integer format that has a 16-bit R component in bytes 0..1, a 16-bit G component in bytes 2..3, and a 16-bit B component in bytes 4..5.
R16G16B16A16_SInt	A four-component, 64-bit signed integer format that has a 16-bit R component in bytes 0..1, a 16-bit G component in bytes 2..3, a 16-bit B component in bytes 4..5, and a 16-bit A component in bytes 6..7.
R32_UInt	A one-component, 32-bit unsigned integer format that has a single 32-bit R component.
R32G32_UInt	A two-component, 64-bit unsigned integer format that has a 32-bit R component in bytes 0..3, and a 32-bit G component in bytes 4..7.
R32G32B32_UInt	A three-component, 96-bit unsigned integer format that has a 32-bit R component in bytes 0..3, a 32-bit G component in bytes 4..7, and a 32-bit B component in bytes 8..11.
R32G32B32A32_UInt	A four-component, 128-bit unsigned integer format that has a 32-bit R component in bytes 0..3, a 32-bit G component in bytes 4..7, a 32-bit B component in bytes 8..11, and a 32-bit A component in bytes 12..15.
R32_SInt	A one-component, 32-bit signed integer format that has a single 32-bit R component.
R32G32_SInt	A two-component, 64-bit signed integer format that has a 32-bit R component in bytes 0..3, and a 32-bit G component in bytes 4..7.
R32G32B32_SInt	A three-component, 96-bit signed integer format that has a 32-bit R component in bytes 0..3, a 32-bit G component in bytes 4..7, and a 32-bit B component in bytes 8..11.

R32G32B32A32_SInt	A four-component, 128-bit signed integer format that has a 32-bit R component in bytes 0..3, a 32-bit G component in bytes 4..7, a 32-bit B component in bytes 8..11, and a 32-bit A component in bytes 12..15.
R16_SFloat	A one-component, 16-bit signed floating-point format that has a single 16-bit R component.
R16G16_SFloat	A two-component, 32-bit signed floating-point format that has a 16-bit R component in bytes 0..1, and a 16-bit G component in bytes 2..3.
R16G16B16_SFloat	A three-component, 48-bit signed floating-point format that has a 16-bit R component in bytes 0..1, a 16-bit G component in bytes 2..3, and a 16-bit B component in bytes 4..5.
R16G16B16A16_SFloat	A four-component, 64-bit signed floating-point format that has a 16-bit R component in bytes 0..1, a 16-bit G component in bytes 2..3, a 16-bit B component in bytes 4..5, and a 16-bit A component in bytes 6..7.
R32_SFloat	A one-component, 32-bit signed floating-point format that has a single 32-bit R component.
R32G32_SFloat	A two-component, 64-bit signed floating-point format that has a 32-bit R component in bytes 0..3, and a 32-bit G component in bytes 4..7.
R32G32B32_SFloat	A three-component, 96-bit signed floating-point format that has a 32-bit R component in bytes 0..3, a 32-bit G component in bytes 4..7, and a 32-bit B component in bytes 8..11.
R32G32B32A32_SFloat	A four-component, 128-bit signed floating-point format that has a 32-bit R component in bytes 0..3, a 32-bit G component in bytes 4..7, a 32-bit B component in bytes 8..11, and a 32-bit A component in bytes 12..15.
B8G8R8_SRGB	A three-component, 24-bit unsigned normalized format that has an 8-bit R component stored with sRGB nonlinear encoding in byte 0, an 8-bit G component stored with sRGB nonlinear encoding in byte 1, and an 8-bit B component stored with sRGB nonlinear encoding in byte 2.
B8G8R8A8_SRGB	A four-component, 32-bit unsigned normalized format that has an 8-bit B component stored with sRGB nonlinear encoding in byte 0, an 8-bit G component stored with sRGB nonlinear encoding in byte 1, an 8-bit R component stored with sRGB nonlinear encoding in byte 2, and an 8-bit A component in byte 3.
B8G8R8_UNorm	A three-component, 24-bit unsigned normalized format that has an 8-bit B component in byte 0, an 8-bit G component in byte 1, and an 8-bit R component in byte 2.
B8G8R8A8_UNorm	A four-component, 32-bit unsigned normalized format that has an 8-bit B component in byte 0, an 8-bit G component in byte 1, an 8-bit R component in byte 2, and an 8-bit A component in byte 3.

B8G8R8_SNorm	A three-component, 24-bit signed normalized format that has an 8-bit B component in byte 0, an 8-bit G component in byte 1, and an 8-bit R component in byte 2.
B8G8R8A8_SNorm	A four-component, 32-bit signed normalized format that has an 8-bit B component in byte 0, an 8-bit G component in byte 1, an 8-bit R component in byte 2, and an 8-bit A component in byte 3.
B8G8R8_UInt	A three-component, 24-bit unsigned integer format that has an 8-bit B component in byte 0, an 8-bit G component in byte 1, and an 8-bit R component in byte 2
B8G8R8A8_UInt	A four-component, 32-bit unsigned integer format that has an 8-bit B component in byte 0, an 8-bit G component in byte 1, an 8-bit R component in byte 2, and an 8-bit A component in byte 3.
B8G8R8_SInt	A three-component, 24-bit signed integer format that has an 8-bit B component in byte 0, an 8-bit G component in byte 1, and an 8-bit R component in byte 2.
B8G8R8A8_SInt	A four-component, 32-bit signed integer format that has an 8-bit B component in byte 0, an 8-bit G component in byte 1, an 8-bit R component in byte 2, and an 8-bit A component in byte 3.
R4G4B4A4_UNormPack16	A four-component, 16-bit packed unsigned normalized format that has a 4-bit R component in bits 12..15, a 4-bit G component in bits 8..11, a 4-bit B component in bits 4..7, and a 4-bit A component in bits 0..3.
B4G4R4A4_UNormPack16	A four-component, 16-bit packed unsigned normalized format that has a 4-bit B component in bits 12..15, a 4-bit G component in bits 8..11, a 4-bit R component in bits 4..7, and a 4-bit A component in bits 0..3.
R5G6B5_UNormPack16	A three-component, 16-bit packed unsigned normalized format that has a 5-bit R component in bits 11..15, a 6-bit G component in bits 5..10, and a 5-bit B component in bits 0..4.
B5G6R5_UNormPack16	A three-component, 16-bit packed unsigned normalized format that has a 5-bit B component in bits 11..15, a 6-bit G component in bits 5..10, and a 5-bit R component in bits 0..4.
R5G5B5A1_UNormPack16	A four-component, 16-bit packed unsigned normalized format that has a 5-bit R component in bits 11..15, a 5-bit G component in bits 6..10, a 5-bit B component in bits 1..5, and a 1-bit A component in bit 0.
B5G5R5A1_UNormPack16	A four-component, 16-bit packed unsigned normalized format that has a 5-bit B component in bits 11..15, a 5-bit G component in bits 6..10, a 5-bit R component in bits 1..5, and a 1-bit A component in bit 0.
A1R5G5B5_UNormPack16	A four-component, 16-bit packed unsigned normalized format that has a 1-bit A component in bit 15, a 5-bit R component in bits 10..14, a 5-bit G

	component in bits 5..9, and a 5-bit B component in bits 0..4.
E5B9G9R9_UFloatPack32	A three-component, 32-bit packed unsigned floating-point format that has a 5-bit shared exponent in bits 27..31, a 9-bit B component mantissa in bits 18..26, a 9-bit G component mantissa in bits 9..17, and a 9-bit R component mantissa in bits 0..8.
B10G11R11_UFloatPack32	A three-component, 32-bit packed unsigned floating-point format that has a 10-bit B component in bits 22..31, an 11-bit G component in bits 11..21, an 11-bit R component in bits 0..10.
A2B10G10R10_UNormPack32	A four-component, 32-bit packed unsigned normalized format that has a 2-bit A component in bits 30..31, a 10-bit B component in bits 20..29, a 10-bit G component in bits 10..19, and a 10-bit R component in bits 0..9.
A2B10G10R10_UIntPack32	A four-component, 32-bit packed unsigned integer format that has a 2-bit A component in bits 30..31, a 10-bit B component in bits 20..29, a 10-bit G component in bits 10..19, and a 10-bit R component in bits 0..9.
A2B10G10R10_SIntPack32	A four-component, 32-bit packed signed integer format that has a 2-bit A component in bits 30..31, a 10-bit B component in bits 20..29, a 10-bit G component in bits 10..19, and a 10-bit R component in bits 0..9.
A2R10G10B10_UNormPack32	A four-component, 32-bit packed unsigned normalized format that has a 2-bit A component in bits 30..31, a 10-bit R component in bits 20..29, a 10-bit G component in bits 10..19, and a 10-bit B component in bits 0..9.
A2R10G10B10_UIntPack32	A four-component, 32-bit packed unsigned integer format that has a 2-bit A component in bits 30..31, a 10-bit R component in bits 20..29, a 10-bit G component in bits 10..19, and a 10-bit B component in bits 0..9.
A2R10G10B10_SIntPack32	A four-component, 32-bit packed signed integer format that has a 2-bit A component in bits 30..31, a 10-bit R component in bits 20..29, a 10-bit G component in bits 10..19, and a 10-bit B component in bits 0..9.
A2R10G10B10_XRSRGBPack32	A four-component, 32-bit packed unsigned normalized format that has a 2-bit A component in bits 30..31, a 10-bit R component in bits 20..29, a 10-bit G component in bits 10..19, and a 10-bit B component in bits 0..9. The components are gamma encoded and their values range from -0.5271 to 1.66894. The alpha component is clamped to either 0.0 or 1.0 on sampling, rendering, and writing operations.
A2R10G10B10_XRUNormPack32	A four-component, 32-bit packed unsigned normalized format that has a 2-bit A component in bits 30..31, a 10-bit R component in bits 20..29, a 10-bit G component in bits 10..19, and a 10-bit B component in bits 0..9. The components are linearly encoded and their values range from -0.752941 to 1.25098 (pre-expansion). The alpha component is clamped to either 0.0 or 1.0 on sampling, rendering, and writing operations.
R10G10B10_XRSRGBPack32	A four-component, 32-bit packed unsigned normalized format that has a

	10-bit R component in bits 20..29, a 10-bit G component in bits 10..19, and a 10-bit B component in bits 0..9. The components are gamma encoded and their values range from -0.5271 to 1.66894. The alpha component is clamped to either 0.0 or 1.0 on sampling, rendering, and writing operations.
R10G10B10_XRUNormPack32	A four-component, 32-bit packed unsigned normalized format that has a 10-bit R component in bits 20..29, a 10-bit G component in bits 10..19, and a 10-bit B component in bits 0..9. The components are linearly encoded and their values range from -0.752941 to 1.25098 (pre-expansion).
A10R10G10B10_XRSRGBPack32	A four-component, 64-bit packed unsigned normalized format that has a 10-bit A component in bits 30..39, a 10-bit R component in bits 20..29, a 10-bit G component in bits 10..19, and a 10-bit B component in bits 0..9. The components are gamma encoded and their values range from -0.5271 to 1.66894. The alpha component is clamped to either 0.0 or 1.0 on sampling, rendering, and writing operations.
A10R10G10B10_XRUNormPack32	A four-component, 64-bit packed unsigned normalized format that has a 10-bit A component in bits 30..39, a 10-bit R component in bits 20..29, a 10-bit G component in bits 10..19, and a 10-bit B component in bits 0..9. The components are linearly encoded and their values range from -0.752941 to 1.25098 (pre-expansion). The alpha component is clamped to either 0.0 or 1.0 on sampling, rendering, and writing operations.
D16_UNorm	A one-component, 16-bit unsigned normalized format that has a single 16-bit depth component.
D24_UNorm	A two-component, 32-bit format that has 24 unsigned normalized bits in the depth component and, optionally: 8 bits that are unused.
D24_UNorm_S8_UInt	A two-component, 32-bit packed format that has 8 unsigned integer bits in the stencil component, and 24 unsigned normalized bits in the depth component.
D32_SFloat	A one-component, 32-bit signed floating-point format that has 32-bits in the depth component.
D32_SFloat_S8_Uint	A two-component format that has 32 signed float bits in the depth component and 8 unsigned integer bits in the stencil component. There are optionally: 24-bits that are unused.
S8_Uint	A one-component, 8-bit unsigned integer format that has 8-bits in the stencil component.
RGBA_DXT1_SRGB	A three-component, block-compressed format. Each 64-bit compressed texel block encodes a 4×4 rectangle of unsigned normalized RGB texel data with sRGB nonlinear encoding. This format has a 1 bit alpha channel.
RGBA_DXT1_UNorm	A three-component, block-compressed format. Each 64-bit compressed

	texel block encodes a 4×4 rectangle of unsigned normalized RGB texel data. This format has a 1 bit alpha channel.
RGBA_DXT3_SRGB	A four-component, block-compressed format where each 128-bit compressed texel block encodes a 4×4 rectangle of unsigned normalized RGBA texel data with the first 64 bits encoding alpha values followed by 64 bits encoding RGB values with sRGB nonlinear encoding.
RGBA_DXT3_UNorm	A four-component, block-compressed format where each 128-bit compressed texel block encodes a 4×4 rectangle of unsigned normalized RGBA texel data with the first 64 bits encoding alpha values followed by 64 bits encoding RGB values.
RGBA_DXT5_SRGB	A four-component, block-compressed format where each 128-bit compressed texel block encodes a 4×4 rectangle of unsigned normalized RGBA texel data with the first 64 bits encoding alpha values followed by 64 bits encoding RGB values with sRGB nonlinear encoding.
RGBA_DXT5_UNorm	A four-component, block-compressed format where each 128-bit compressed texel block encodes a 4×4 rectangle of unsigned normalized RGBA texel data with the first 64 bits encoding alpha values followed by 64 bits encoding RGB values.
R_BC4_UNorm	A one-component, block-compressed format where each 64-bit compressed texel block encodes a 4×4 rectangle of unsigned normalized red texel data.
R_BC4_SNorm	A one-component, block-compressed format where each 64-bit compressed texel block encodes a 4×4 rectangle of signed normalized red texel data.
RG_BC5_UNorm	A two-component, block-compressed format where each 128-bit compressed texel block encodes a 4×4 rectangle of unsigned normalized RG texel data with the first 64 bits encoding red values followed by 64 bits encoding green values.
RG_BC5_SNorm	A two-component, block-compressed format where each 128-bit compressed texel block encodes a 4×4 rectangle of signed normalized RG texel data with the first 64 bits encoding red values followed by 64 bits encoding green values.
RGB_BC6H_UFloat	A three-component, block-compressed format where each 128-bit compressed texel block encodes a 4×4 rectangle of unsigned floating-point RGB texel data.
RGB_BC6H_SFloat	A three-component, block-compressed format where each 128-bit compressed texel block encodes a 4×4 rectangle of signed floating-point RGB texel data.
RGBA_BC7_SRGB	A four-component, block-compressed format where each 128-bit

	compressed texel block encodes a 4×4 rectangle of unsigned normalized RGBA texel data with sRGB nonlinear encoding applied to the RGB components.
RGBA_BC7_UNorm	A four-component, block-compressed format where each 128-bit compressed texel block encodes a 4×4 rectangle of unsigned normalized RGBA texel data.
RGB_PVRTC_2Bpp_SRGB	A three-component, PVRTC compressed format where each 64-bit compressed texel block encodes a 8×4 rectangle of unsigned normalized RGB texel data with sRGB nonlinear encoding. This format has no alpha and is considered opaque.
RGB_PVRTC_2Bpp_UNorm	A three-component, PVRTC compressed format where each 64-bit compressed texel block encodes a 8×4 rectangle of unsigned normalized RGB texel data. This format has no alpha and is considered opaque.
RGB_PVRTC_4Bpp_SRGB	A three-component, PVRTC compressed format where each 64-bit compressed texel block encodes a 4×4 rectangle of unsigned normalized RGB texel data with sRGB nonlinear encoding. This format has no alpha and is considered opaque.
RGB_PVRTC_4Bpp_UNorm	A three-component, PVRTC compressed format where each 64-bit compressed texel block encodes a 4×4 rectangle of unsigned normalized RGB texel data. This format has no alpha and is considered opaque.
RGBA_PVRTC_2Bpp_SRGB	A four-component, PVRTC compressed format where each 64-bit compressed texel block encodes a 8×4 rectangle of unsigned normalized RGBA texel data with the first 32 bits encoding alpha values followed by 32 bits encoding RGB values with sRGB nonlinear encoding applied.
RGBA_PVRTC_2Bpp_UNorm	A four-component, PVRTC compressed format where each 64-bit compressed texel block encodes a 8×4 rectangle of unsigned normalized RGBA texel data with the first 32 bits encoding alpha values followed by 32 bits encoding RGB values.
RGBA_PVRTC_4Bpp_SRGB	A four-component, PVRTC compressed format where each 64-bit compressed texel block encodes a 4×4 rectangle of unsigned normalized RGBA texel data with the first 32 bits encoding alpha values followed by 32 bits encoding RGB values with sRGB nonlinear encoding applied.
RGBA_PVRTC_4Bpp_UNorm	A four-component, PVRTC compressed format where each 64-bit compressed texel block encodes a 4×4 rectangle of unsigned normalized RGBA texel data with the first 32 bits encoding alpha values followed by 32 bits encoding RGB values.
RGB_ETC_UNorm	A three-component, ETC compressed format where each 64-bit compressed texel block encodes a 4×4 rectangle of unsigned normalized RGB texel data. This format has no alpha and is considered opaque.

RGB_ETC2_SRGB	A three-component, ETC2 compressed format where each 64-bit compressed texel block encodes a 4×4 rectangle of unsigned normalized RGB texel data with sRGB nonlinear encoding. This format has no alpha and is considered opaque.
RGB_ETC2_UNorm	A three-component, ETC2 compressed format where each 64-bit compressed texel block encodes a 4×4 rectangle of unsigned normalized RGB texel data. This format has no alpha and is considered opaque.
RGB_A1_ETC2_SRGB	A four-component, ETC2 compressed format where each 64-bit compressed texel block encodes a 4×4 rectangle of unsigned normalized RGB texel data with sRGB nonlinear encoding, and provides 1 bit of alpha.
RGB_A1_ETC2_UNorm	A four-component, ETC2 compressed format where each 64-bit compressed texel block encodes a 4×4 rectangle of unsigned normalized RGB texel data, and provides 1 bit of alpha.
RGBA_ETC2_SRGB	A four-component, ETC2 compressed format where each 128-bit compressed texel block encodes a 4×4 rectangle of unsigned normalized RGBA texel data with the first 64 bits encoding alpha values followed by 64 bits encoding RGB values with sRGB nonlinear encoding applied.
RGBA_ETC2_UNorm	A four-component, ETC2 compressed format where each 128-bit compressed texel block encodes a 4×4 rectangle of unsigned normalized RGBA texel data with the first 64 bits encoding alpha values followed by 64 bits encoding RGB values.
R_EAC_UNorm	A one-component, ETC2 compressed format where each 64-bit compressed texel block encodes a 4×4 rectangle of unsigned normalized red texel data.
R_EAC_SNorm	A one-component, ETC2 compressed format where each 64-bit compressed texel block encodes a 4×4 rectangle of signed normalized red texel data.
RG_EAC_UNorm	A two-component, ETC2 compressed format where each 128-bit compressed texel block encodes a 4×4 rectangle of unsigned normalized RG texel data with the first 64 bits encoding red values followed by 64 bits encoding green values.
RG_EAC_SNorm	A two-component, ETC2 compressed format where each 128-bit compressed texel block encodes a 4×4 rectangle of signed normalized RG texel data with the first 64 bits encoding red values followed by 64 bits encoding green values.
RGBA_ASTC4X4_SRGB	A four-component, ASTC compressed format where each 128-bit compressed texel block encodes a 4×4 rectangle of unsigned normalized RGBA texel data with sRGB nonlinear encoding applied to the RGB components.

RGBA_ASTC4X4_UNorm	A four-component, ASTC compressed format where each 128-bit compressed texel block encodes a 4×4 rectangle of unsigned normalized RGBA texel data.
RGBA_ASTC5X5_SRGB	A four-component, ASTC compressed format where each 128-bit compressed texel block encodes a 5×5 rectangle of unsigned normalized RGBA texel data with sRGB nonlinear encoding applied to the RGB components.
RGBA_ASTC5X5_UNorm	A four-component, ASTC compressed format where each 128-bit compressed texel block encodes a 5×5 rectangle of unsigned normalized RGBA texel data.
RGBA_ASTC6X6_SRGB	A four-component, ASTC compressed format where each 128-bit compressed texel block encodes a 6×6 rectangle of unsigned normalized RGBA texel data with sRGB nonlinear encoding applied to the RGB components.
RGBA_ASTC6X6_UNorm	A four-component, ASTC compressed format where each 128-bit compressed texel block encodes a 6×6 rectangle of unsigned normalized RGBA texel data.
RGBA_ASTC8X8_SRGB	A four-component, ASTC compressed format where each 128-bit compressed texel block encodes an 8×8 rectangle of unsigned normalized RGBA texel data with sRGB nonlinear encoding applied to the RGB components.
RGBA_ASTC8X8_UNorm	A four-component, ASTC compressed format where each 128-bit compressed texel block encodes an 8×8 rectangle of unsigned normalized RGBA texel data.
RGBA_ASTC10X10_SRGB	A four-component, ASTC compressed format where each 128-bit compressed texel block encodes a 10×10 rectangle of unsigned normalized RGBA texel data with sRGB nonlinear encoding applied to the RGB components.
RGBA_ASTC10X10_UNorm	A four-component, ASTC compressed format where each 128-bit compressed texel block encodes a 10×10 rectangle of unsigned normalized RGBA texel data.
RGBA_ASTC12X12_SRGB	A four-component, ASTC compressed format where each 128-bit compressed texel block encodes a 12×12 rectangle of unsigned normalized RGBA texel data with sRGB nonlinear encoding applied to the RGB components.
RGBA_ASTC12X12_UNorm	A four-component, ASTC compressed format where each 128-bit compressed texel block encodes a 12×12 rectangle of unsigned normalized RGBA texel data.

Description

Visible reflection probes sorting options.

See Also: CullingParameters.reflectionProbeSortOptions.

Properties

None	Do not sort reflection probes.
Importance	Sort probes by importance.
Size	Sort probes from largest to smallest.
ImportanceThenSize	Sort probes by importance, then by size.

Description

What kind of per-object data to setup during rendering.

At minimum, object transformation matrices are set up per-object, but extra data (e.g. lightmaps, light probes etc.) can be set up by combining RendererConfiguration flags.

See Also: DrawRendererSettings, RenderLoop.DrawRenderers.

Properties

None	Do not setup any particular per-object data besides the transformation matrix.
PerObjectLightProbe	Setup per-object light probe SH data.
PerObjectReflectionProbes	Setup per-object reflection probe data.
PerObjectLightProbeProxyVolume	Setup per-object light probe proxy volume data.
PerObjectLightmaps	Setup per-object lightmaps.
ProvideLightIndices	Setup per-object light indices.
PerObjectMotionVectors	Setup per-object motion vectors.
PerObjectOcclusionProbe	Setup per-object occlusion probe data.
PerObjectOcclusionProbeProxyVolume	Setup per-object occlusion probe proxy volume data (occlusion in alpha channels).
PerObjectShadowMask	Setup per-object shadowmask.

Description

Specifies which parts of the render state that is overriden.

In order to override multiple parts of the render state, the desired values should be logically OR'd together.

See Also: RenderStateBlock.

Properties

Nothing	No render states are overridden.
Blend	When set, the blend state is overridden.
Raster	When set, the raster state is overridden.
Depth	When set, the depth state is overridden.
Stencil	When set, the stencil state and reference value is overridden.
Everything	When set, all render states are overridden.

Description

How to sort objects during rendering.

Control the way Unity sorts objects before drawing them by using and combining these flags.

The basic flags are:
- SortingLayer,
- RenderQueue,
- BackToFront,
- QuantizedFrontToBack,
- OptimizeStateChanges,
- CanvasOrder.

Multiple flags, when combined, are applied in the above order.

Some commonly-used sorting combinations are provided for convenience. Use CommonOpaque for opaque objects. This combination of flags includes optimization for reducing draw state changes and draws roughly front-to-back to reduce drawing over the same pixels many times. Use CommonTransparent for transparent objects, which need to be sorted from back to front before being drawn for them all to be visible.

See Also: DrawRendererSettings.sorting, RenderLoop.DrawRenderers.

Properties

None	Do not sort objects.
SortingLayer	Sort by renderer sorting layer.
RenderQueue	Sort by material render queue.
BackToFront	Sort objects back to front.
QuantizedFrontToBack	Sort objects in rough front-to-back buckets.
OptimizeStateChanges	Sort objects to reduce draw state changes.
CanvasOrder	Sort renderers taking canvas order into account.
RendererPriority	Sorts objects by renderer priority.
CommonOpaque	Typical sorting for opaque objects.
CommonTransparent	Typical sorting for transparencies.

Description

Flags for VisibleLight.

See Also: VisibleLight.flags, CullResults.visibleLights.

Properties

None	No flags are set.
IntersectsNearPlane	Light intersects near clipping plane.
IntersectsFarPlane	Light intersects far clipping plane.

Description

Describes the information about the edge and how to tessellate it.

Properties

end	The maximum angle to be considered within this range.
order	The render order of the edges that belong in this range.
sprites	The list of Sprites that are associated with this range.
start	The minimum angle to be considered within this range.

Description

A collection of APIs that facilitate pixel perfect rendering of sprite-based renderers.

Static Properties

pixelSnapSpacing	To achieve a pixel perfect render, Sprites must be displaced to discrete positions at render time. This value defines the minimum distance between these positions. This doesn't affect the GameObject's transform position.

Description

Data that describes the important points of the shape.

Properties

leftTangent	The position of the left tangent in local space.
mode	The various modes of the tangent handles. They could be continuous or broken.
position	The position of this point in the object's local space.
rightTangent	The position of the right tangent point in the local space.

Description

A struct that holds a rich set of information that describes the bind pose of this Sprite.

Most importantly this struct describes the hierarchy and various other spatial relationships between the bones.

Properties

length	The length of the bone. This is important for the leaf bones to describe their length without needing another bone as the terminal bone.
name	The name of the bone. This is useful when recreating bone hierarchy at editor or runtime. You can also use this as a way of resolving the bone path when a Sprite is bound to a more complex or richer hierarchy.
parentId	The ID of the parent of this bone.
position	The position in local space of this bone.
rotation	The rotation of this bone in local space.

Description

A list of methods designed for reading and writing to the rich internal data of a Sprite.

It is now possible to read and write to all the channels of the VertexAttribute, BoneWeight, BindPose and the SpriteBones of a Sprite.

Static Methods

GetBindPoses	Returns an array of BindPoses.
GetBones	Returns a list of SpriteBone in this Sprite.
GetBoneWeights	Returns a list of BoneWeight that corresponds to each and every vertice in this Sprite.
GetIndices	Returns a list of indices. This is the same as Sprite.triangle.
GetVertexAttribute	Retrieves a strided accessor to the internal vertex attributes.
GetVertexCount	Returns the number of vertices in this Sprite.
HasVertexAttribute	Checks if a specific channel exists for this Sprite.
SetBindPoses	Sets the bind poses for this Sprite.
SetBones	Sets the SpriteBones for this Sprite.
SetBoneWeights	Sets the BoneWeight for this Sprite. The length of the input array must match the number of vertices.
SetIndices	Set the indices for this Sprite. This is the same as Sprite.triangle.
SetVertexAttribute	Sets a specific channel of the VertexAttribute.
SetVertexCount	Sets the vertex count. This resizes the internal buffer. It also preserves any configurations of VertexAttributes.

Description

A list of methods that allow the caller to override what the SpriteRenderer renders.

This feature is used mainly in the Sprite Animation feature. The SpriteSkin component gets the deformable buffer and deforms it using the skin weights and bones. After the deformation is complete, the Sprite Renderer uses it to render the Sprite instead of the Sprite's original mesh.

Static Methods

DeactivateDeformableBuffer	Stop using the deformable buffer to render the Sprite and use the original mesh instead.
GetDeformableVertices	Returns an array of vertices to be deformed by the caller.
UpdateDeformableBuffer	Provides the JobHandle that updates the deform buffer to the SpriteRenderer.

Description

Additional data about the shape's control point. This is useful during tessellation of the shape.

Properties

bevelCutoff	The threshold of the angle that decides if it should be tessellated as a curve or a corner.
bevelSize	The radius of the curve to be tessellated.
corner	True will indicate that this point should be tessellated as a corner or a continuous line otherwise.
height	The height of the tessellated edge.
spriteIndex	The Sprite to be used for a particular edge.

Description

Input parameters for the SpriteShape tessellator.

Properties

adaptiveUV	If enabled, the tessellator will adapt the size of the quads based on the height of the edge.
angleThreshold	The threshold of the angle that indicates whether it is a corner or not.
bevelCutoff	The threshold of the angle that decides if it should be tessellated as a curve or a corner.
bevelSize	The radius of the curve to be tessellated.
borderPivot	The local displacement of the Sprite when tessellated.
carpet	If true, the Shape will be tessellated as a closed form.
fillScale	The scale to be used to calculate the UVs of the fill texture.
fillTexture	The texture to be used for the fill of the SpriteShape.
smartSprite	If enabled the tessellator will consider creating corners based on the various input parameters.
splineDetail	The tessellation quality of the input Spline that determines the complexity of the mesh.
spriteBorders	The borders to be used for calculating the uv of the edges based on the border info found in Sprites.
transform	The world space transform of the game object used for calculating the UVs of the fill texture.

Description

Renders SpriteShapes defined through the SpriteShapeUtility.GenerateSpriteShape API.

Inherited Members

Properties

gameObject	The game object this component is attached to. A component is always attached to a game object.
tag	The tag of this game object.
transform	The Transform attached to this GameObject.
hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.
allowOcclusionWhenDynamic	Controls if dynamic occlusion culling should be performed for this renderer.
bounds	The bounding volume of the renderer (Read Only).
enabled	Makes the rendered 3D object visible if enabled.
isPartOfStaticBatch	Has this renderer been statically batched with any other renderers?
isVisible	Is this renderer visible in any camera? (Read Only)
lightmapIndex	The index of the baked lightmap applied to this renderer.
lightmapScaleOffset	The UV scale & offset used for a lightmap.
lightProbeProxyVolumeOverride	If set, the Renderer will use the Light Probe Proxy Volume component attached to the source GameObject.
lightProbeUsage	The light probe interpolation type.
localToWorldMatrix	Matrix that transforms a point from local space into world space (Read Only).
material	Returns the first instantiated Material assigned to the renderer.
materials	Returns all the instantiated materials of this object.
motionVectorGenerationMode	Specifies the mode for motion vector rendering.
probeAnchor	If set, Renderer will use this Transform's position to find the light or

	reflection probe.
realtimeLightmapIndex	The index of the realtime lightmap applied to this renderer.
realtimeLightmapScaleOffset	The UV scale & offset used for a realtime lightmap.
receiveShadows	Does this object receive shadows?
reflectionProbeUsage	Should reflection probes be used for this Renderer?
rendererPriority	This value sorts renderers by priority. Lower values are rendered first and higher values are rendered last.
renderingLayerMask	Determines which rendering layer this renderer lives on.
shadowCastingMode	Does this object cast shadows?
sharedMaterial	The shared material of this object.
sharedMaterials	All the shared materials of this object.
sortingLayerID	Unique ID of the Renderer's sorting layer.
sortingLayerName	Name of the Renderer's sorting layer.
sortingOrder	Renderer's order within a sorting layer.
worldToLocalMatrix	Matrix that transforms a point from world space into local space (Read Only).

Public Methods

BroadcastMessage	Calls the method named methodName on every MonoBehaviour in this game object or any of its children.
CompareTag	Is this game object tagged with tag ?
GetComponent	Returns the component of Type type if the game object has one attached, null if it doesn't.
GetComponentInChildren	Returns the component of Type type in the GameObject or any of its children using depth first search.
GetComponentInParent	Returns the component of Type type in the GameObject or any of its parents.
GetComponents	Returns all components of Type type in the GameObject.
GetComponentsInChildren	Returns all components of Type type in the GameObject or any of its children.
GetComponentsInParent	Returns all components of Type type in the GameObject or any of its parents.
SendMessage	Calls the method named methodName on every MonoBehaviour in this game object.

SendMessageUpwards	Calls the method named methodName on every MonoBehaviour in this game object and on every ancestor of the behaviour.
GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.
GetClosestReflectionProbes	Returns an array of closest reflection probes with weights, weight shows how much influence the probe has on the renderer, this value is also used when blending between reflection probes occur.
GetMaterials	Returns all the instantiated materials of this object.
GetPropertyBlock	Get per-Renderer or per-Material property block.
GetSharedMaterials	Returns all the shared materials of this object.
HasPropertyBlock	Returns true if the Renderer has a material property block attached via SetPropertyBlock.
SetPropertyBlock	Lets you set or clear per-renderer or per-material parameter overrides.

Static Methods

Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Messages

OnBecameInvisible	OnBecameInvisible is called when the object is no longer visible by any camera.

OnBecameVisible	OnBecameVisible is called when the object became visible by any camera.

Description

A static class that helps tessellate a SpriteShape mesh.

Static Methods

Generate	Generate a mesh based on input parameters.
GenerateSpriteShape	Generate a mesh based on input parameters.

Description

This enumeration contains values to control how an element is aligned in its parent during the layout phase.

The cross axis of a container is perpendicular to the primary axis on which children are laid out, that is, its flex direction.

Properties

Auto	Default value (currently FlexStart).
FlexStart	Items are aligned at the beginning on the cross axis.
Center	Items are centered on the cross axis.
FlexEnd	Items are aligned at the end on the cross axis.
Stretch	Stretches items on the cross axis.

Description

This enumeration defines values used to control in which direction a container will place its children during layout.

Properties

Column	Vertical layout.
ColumnReverse	Vertical layout in reverse order.
Row	Horizontal layout.
RowReverse	Horizontal layout in reverse order.

Description

This enumeration contains values to control how children are justified during layout.

The main axis of a container is the primary axis on which children are laid out, that is, its flex direction.

Properties

FlexStart	Items are justified towards the beginning of the main axis.
Center	Items are centered.
FlexEnd	Items are justified towards the end of the layout direction.
SpaceBetween	Items are evenly distributed in the line; first item is at the beginning of the line, last item is at the end.
SpaceAround	Items are evenly distributed in the line with extra space on each end of the line.

Description

This enumeration contains values to control how an element is positioned in its parent container.

Properties

Relative	The element is positioned in relation to its default box as calculated by layout.
Absolute	The element is positioned in relation to its parent box and does not contribute to the layout anymore.

Description

This enumeration contains values to specify whether or not an element is visible.

Properties

Visible	The element is drawn normally (default).
Hidden	The picking and rendering of this element is skipped. It still takes space in the layout.

Description

This enumeration contains values to control how elements are placed in a container if not enough space is available.

Properties

NoWrap	All elements are placed on the same line.
Wrap	Elements are placed over multiple lines.
WrapReverse	Elements are placed over multiple lines with new lines occuring in the reverse order set in the container.

Description

This generic structure encodes a value type that can come from USS or be specified programmatically.

A StyleValue<T> struct wraps value types such as an integer or a color along with some metadata for the application of USS at runtime.

Static Properties

nil	This represents the default value for a StyleValue<T> of the according generic type.

Properties

value	The actual value of the StyleValue<T>.

Constructors

StyleValue_1	This constructor can be used to specified an alternate default value but it is recommended to use StyleValue<T>.nil.

Public Methods

GetSpecifiedValueOrDefault	Utility function to be used when reading custom styles values and provide a default value in one step.

Static Methods

Create	Creates a StyleValue of the according generic type directly from a value.

Description

This interface exposes methods to read custom style properties applied from USS files to visual elements.

Public Methods

ApplyCustomProperty	Read a style property value into the specified StyleValue<T>.

Description

Event sent after an element is added to an element that is a descendent of a panel.

Constructors

AttachToPanelEvent	Constructor. Avoid newing events. Instead, use GetPooled() to get an event from a pool of reusable events.

Inherited Members

Properties

bubbles	Whether this event type bubbles up in the event propagation path.
currentTarget	The current target of the event. The current target is the element in the propagation path for which event handlers are currently being executed.
dispatch	Whether the event is being dispatched to a visual element. An event cannot be redispatched while it being dispatched. If you need to recursively dispatch an event, it is recommended that you use a copy of the event.
flags	Flags for the event.
imguiEvent	The IMGUIEvent at the source of this event. The source can be null since not all events are generated by IMGUI.
isDefaultPrevented	Return true if the default actions should not be executed for this event.
isImmediatePropagationStopped	Whether StopImmediatePropagation() was called for this event.
isPropagationStopped	Whether StopPropagation() was called for this event.
originalMousePosition	The original mouse position of the IMGUI event, before it is transformed to the current target local coordinates.
pooled	Whether the event is allocated from a pool of events.
propagationPhase	The current propagation phase.
target	The target visual element that received this event. Unlike currentTarget, this target does not change when the event is sent to other elements along the propagation path.
timestamp	The time when the event was created.

tricklesDown	Whether this event is sent down the event propagation path during the TrickleDown phase.
destinationPanel	In the case of AttachToPanelEvent, the panel to which the event target element is now attached. In the case of DetachFromPanelEvent, the panel to which the event target element will be attached.
originPanel	In the case of AttachToPanelEvent, the panel to which the event target element was attached. In the case of DetachFromPanelEvent, the panel from which the event target element is detached.

Public Methods

Dispose	Implementation of IDisposable.
GetEventTypeId	Retrieves the type id for this event instance.
PreventDefault	Whether the default actions are prevented from being executed for this event.
StopImmediatePropagation	Immediately stops the propagation of the event. The event is not sent to other elements along the propagation path. This method prevents other event handlers from executing on the current target.
StopPropagation	Stops propagating this event. The event is not sent to other elements along the propagation path. This method does not prevent other event handlers from executing on the current target.
Dispose	Implementation of IDispose.
GetEventTypeId	Retrieves the type id for this event instance.

Protected Methods

Init	Resets all event members to their initial values.
Init	Resets all event members to their initial values.
Init	Resets the event members to their initial values.

Static Methods

RegisterEventType	Registers an event class to the event type system.
GetPooled	Gets an event from the event pool. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().
TypeId	Gets the type id for the event class.
GetPooled	Gets an event from the event pool and initializes it with the given values. Use this

function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().

Description

Event sent immediately after an element has lost focus. This event trickles down, it does not bubble up, and it cannot be cancelled.

Constructors

BlurEvent	Constructor. Avoid newing events. Instead, use GetPooled() to get an event from a pool of reusable events.

Inherited Members

Properties

bubbles	Whether this event type bubbles up in the event propagation path.
currentTarget	The current target of the event. The current target is the element in the propagation path for which event handlers are currently being executed.
dispatch	Whether the event is being dispatched to a visual element. An event cannot be redispatched while it being dispatched. If you need to recursively dispatch an event, it is recommended that you use a copy of the event.
flags	Flags for the event.
imguiEvent	The IMGUIEvent at the source of this event. The source can be null since not all events are generated by IMGUI.
isDefaultPrevented	Return true if the default actions should not be executed for this event.
isImmediatePropagationStopped	Whether StopImmediatePropagation() was called for this event.
isPropagationStopped	Whether StopPropagation() was called for this event.
originalMousePosition	The original mouse position of the IMGUI event, before it is transformed to the current target local coordinates.
pooled	Whether the event is allocated from a pool of events.
propagationPhase	The current propagation phase.
target	The target visual element that received this event. Unlike currentTarget, this target does not change when the event is sent to other elements along the propagation path.
timestamp	The time when the event was created.

tricklesDown	Whether this event is sent down the event propagation path during the TrickleDown phase.
direction	Direction of the focus change.
relatedTarget	For FocusOut and Blur events, contains the element that gains the focus. For FocusIn and Focus events, contains the element that loses the focus.

Public Methods

Dispose	Implementation of IDisposable.
GetEventTypeId	Retrieves the type id for this event instance.
PreventDefault	Whether the default actions are prevented from being executed for this event.
StopImmediatePropagation	Immediately stops the propagation of the event. The event is not sent to other elements along the propagation path. This method prevents other event handlers from executing on the current target.
StopPropagation	Stops propagating this event. The event is not sent to other elements along the propagation path. This method does not prevent other event handlers from executing on the current target.
Dispose	Implementation of IDispose.
GetEventTypeId	Retrieves the type id for this event instance.

Protected Methods

Init	Resets all event members to their initial values.
Init	Resets all event members to their initial values.
Init	Resets the event members to their initial values.

Static Methods

RegisterEventType	Registers an event class to the event type system.
GetPooled	Gets an event from the event pool. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().
TypeId	Gets the type id for the event class.
GetPooled	Gets an event from the event pool and initializes the event with the given values. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().

Description

Styled visual element to match the IMGUI Box Style.

Inherited Members

Static Properties

defaultFocusIndex	The default focus index for newly created elements.

Properties

canGrabFocus	Return true if the element can be focused.
focusController	Return the focus controller for this element.
focusIndex	An integer used to sort focusables in the focus ring. A negative value means that the element can not be focused.
childCount	Number of child elements in this object's contentContainer
clippingOptions	Should this element clip painting to its boundaries.
contentContainer	child elements are added to this element, usually this
dataWatch	Access to this element data watch interface.
enabledInHierarchy	Returns true if the VisualElement is enabled in its own hierarchy.
enabledSelf	Returns true if the VisualElement is enabled locally.
persistenceKey	Used for view data persistence (ie. tree expanded states, scroll position, zoom level).
schedule	Retrieves this VisualElement's IVisualElementScheduler
shadow	Access to this element physical hierarchy
style	Reference to the style object of this element.
this[int]	Access to this element physical hierarchy
tooltip	Text to display inside an information box after the user hovers the element for a small amount of time.
userData	This property can be used to associate application-specific user data with this VisualElement.

Public Methods

HandleEvent	Handle an event, most often by executing the callbacks associated with the event.
HasBubbleUpHandlers	Return true if event handlers for the event propagation BubbleUp phase have been attached on this object.
HasTrickleDownHandlers	Returns true if event handlers, for the event propagation TrickleDown phase, are attached to this object.
RegisterCallback	Adds an event handler to the instance. If the event handler has already been registered for the same phase (either TrickleDown or BubbleUp) then this method has no effect.
SendEvent	Sends an event to the event handler.
UnregisterCallback	Remove callback from the instance.
Blur	Tell the element to release the focus.
Focus	Attempt to give the focus to this element.
Add	Add an element to this element's contentContainer
AddStyleSheetPath	Adds this stylesheet file to this element list of applied styles
BringToFront	Brings this element to the end of its parent children list. The element will be visually in front of any overlapping sibling elements.
Children	Returns the elements from its contentContainer
Clear	Remove all child elements from this element's contentContainer
Contains	Returns true if the element is a direct child of this VisualElement
ElementAt	Retrieves the child element at position
EnableInClassList	Enables or disables the class with the given name.
FindAncestorUserData	Searchs up the hierachy of this VisualElement and retrieves stored userData, if any is found.
FindCommonAncestor	Finds the lowest commont ancestor between two VisualElements inside the VisualTree hierarchy
GetEnumerator	Allows to iterate into this elements children
GetFirstAncestorOfType	Walks up the hierarchy, starting from this element's parent, and returns the first VisualElement of this type
GetFirstOfType	Walks up the hierarchy, starting from this element, and returns the first VisualElement of this type
GetFullHierarchicalPersistenceKey	Combine this VisualElement's VisualElement.persistenceKey with those of

	its parents to create a more unique key for use with VisualElement.GetOrCreatePersistentData.
GetOrCreatePersistentData	Takes a reference to an existing persisted object and a key and returns the object either filled with the persisted state or as-is.
HasStyleSheetPath	Checks if this stylesheet file is in this element list of applied styles
IndexOf	Retrieves the child index of the specified VisualElement.
Insert	Insert an element into this element's contentContainer
MarkDirtyRepaint	Triggers a repaint of the VisualElement on the next frame.
OnPersistentDataReady	Called when the persistent data is accessible and/or when the data or persistence key have changed (VisualElement is properly parented).
OverwriteFromPersistedData	Overwrite object from the persistent data store.
PlaceBehind	Places this element right before the sibling element in their parent children list. If the element and the sibling position overlap, the element will be visually behind of its sibling.
PlaceInFront	Places this element right after the sibling element in their parent children list. If the element and the sibling position overlap, the element will be visually in front of its sibling.
Remove	Removes this child from the hierarchy
RemoveAt	Remove the child element located at this position from this element's contentContainer
RemoveFromHierarchy	Removes this element from its parent hierarchy
RemoveStyleSheetPath	Removes this stylesheet file from this element list of applied styles
SavePersistentData	Write persistence data to file.
SendEvent	Sends an event to the event handler.
SendToBack	Sends this element to the beginning of its parent children list. The element will be visually behind any overlapping sibling elements.
SetEnabled	Changes whether the current VisualElement is enabled or not. When disabled, a VisualElement does not receive most events.
Sort	Reorders child elements from this VisualElement contentContainer.
ToggleInClassList	Toggles between adding and removing the given class name from the class list.

Protected Methods

OnStyleResolved	Callback when the styles of an object have changed.

Implements interfaces:IEventHandler

Description

Interface for classes capable of having callbacks to handle events.

Public Methods

HandleEvent	Handle an event, most often by executing the callbacks associated with the event.
HasBubbleUpHandlers	Return true if event handlers for the event propagation BubbleUp phase have been attached on this object.
HasTrickleDownHandlers	Returns true if event handlers, for the event propagation TrickleDown phase, are attached to this object.
RegisterCallback	Adds an event handler to the instance. If the event handler has already been registered for the same phase (either TrickleDown or BubbleUp) then this method has no effect.
SendEvent	Sends an event to the event handler.
UnregisterCallback	Remove callback from the instance.

Implements interfaces:IChangeEvent

Description

Sends an event when a value in a field changes.

Properties

newValue	The new value.
previousValue	The value before the change occured.

Constructors

ChangeEvent_1	Constructor.

Protected Methods

Init	Sets the event to its initial state.

Static Methods

GetPooled	Gets an event from the event pool and initializes it with the given values. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().

Inherited Members

Properties

bubbles	Whether this event type bubbles up in the event propagation path.
currentTarget	The current target of the event. The current target is the element in the propagation path for which event handlers are currently being executed.
dispatch	Whether the event is being dispatched to a visual element. An event cannot be redispatched while it being dispatched. If you need to recursively dispatch an event, it is recommended that you use a copy of the event.
flags	Flags for the event.
imguiEvent	The IMGUIEvent at the source of this event. The source can be null since

	not all events are generated by IMGUI.
isDefaultPrevented	Return true if the default actions should not be executed for this event.
isImmediatePropagationStopped	Whether StopImmediatePropagation() was called for this event.
isPropagationStopped	Whether StopPropagation() was called for this event.
originalMousePosition	The original mouse position of the IMGUI event, before it is transformed to the current target local coordinates.
pooled	Whether the event is allocated from a pool of events.
propagationPhase	The current propagation phase.
target	The target visual element that received this event. Unlike currentTarget, this target does not change when the event is sent to other elements along the propagation path.
timestamp	The time when the event was created.
tricklesDown	Whether this event is sent down the event propagation path during the TrickleDown phase.

Public Methods

Dispose	Implementation of IDisposable.
GetEventTypeId	Retrieves the type id for this event instance.
PreventDefault	Whether the default actions are prevented from being executed for this event.
StopImmediatePropagation	Immediately stops the propagation of the event. The event is not sent to other elements along the propagation path. This method prevents other event handlers from executing on the current target.
StopPropagation	Stops propagating this event. The event is not sent to other elements along the propagation path. This method does not prevent other event handlers from executing on the current target.
Dispose	Implementation of IDispose.
GetEventTypeId	Retrieves the type id for this event instance.

Static Methods

RegisterEventType	Registers an event class to the event type system.
GetPooled	Gets an event from the event pool. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().

TypeId	Gets the type id for the event class.

Implements interfaces:ICommandEvent

Description

Base class for command events.

Properties

commandName	Name of the command.

Protected Methods

Init	Resets the event members to their initial values.

Static Methods

GetPooled	Gets an event from the event pool and initializes it with the given values. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().

Inherited Members

Properties

bubbles	Whether this event type bubbles up in the event propagation path.
currentTarget	The current target of the event. The current target is the element in the propagation path for which event handlers are currently being executed.
dispatch	Whether the event is being dispatched to a visual element. An event cannot be redispatched while it being dispatched. If you need to recursively dispatch an event, it is recommended that you use a copy of the event.
flags	Flags for the event.
imguiEvent	The IMGUIEvent at the source of this event. The source can be null since not all events are generated by IMGUI.
isDefaultPrevented	Return true if the default actions should not be executed for this event.
isImmediatePropagationStopped	Whether StopImmediatePropagation() was called for this event.
isPropagationStopped	Whether StopPropagation() was called for this event.

originalMousePosition	The original mouse position of the IMGUI event, before it is transformed to the current target local coordinates.
pooled	Whether the event is allocated from a pool of events.
propagationPhase	The current propagation phase.
target	The target visual element that received this event. Unlike currentTarget, this target does not change when the event is sent to other elements along the propagation path.
timestamp	The time when the event was created.
tricklesDown	Whether this event is sent down the event propagation path during the TrickleDown phase.

Public Methods

Dispose	Implementation of IDisposable.
GetEventTypeId	Retrieves the type id for this event instance.
PreventDefault	Whether the default actions are prevented from being executed for this event.
StopImmediatePropagation	Immediately stops the propagation of the event. The event is not sent to other elements along the propagation path. This method prevents other event handlers from executing on the current target.
StopPropagation	Stops propagating this event. The event is not sent to other elements along the propagation path. This method does not prevent other event handlers from executing on the current target.
Dispose	Implementation of IDispose.
GetEventTypeId	Retrieves the type id for this event instance.

Static Methods

RegisterEventType	Registers an event class to the event type system.
GetPooled	Gets an event from the event pool. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().
TypeId	Gets the type id for the event class.

Description

The event sent when clicking the right mouse button.

Constructors

ContextClickEvent	Constructor.

Inherited Members

Properties

bubbles	Whether this event type bubbles up in the event propagation path.
currentTarget	The current target of the event. The current target is the element in the propagation path for which event handlers are currently being executed.
dispatch	Whether the event is being dispatched to a visual element. An event cannot be redispatched while it being dispatched. If you need to recursively dispatch an event, it is recommended that you use a copy of the event.
flags	Flags for the event.
imguiEvent	The IMGUIEvent at the source of this event. The source can be null since not all events are generated by IMGUI.
isDefaultPrevented	Return true if the default actions should not be executed for this event.
isImmediatePropagationStopped	Whether StopImmediatePropagation() was called for this event.
isPropagationStopped	Whether StopPropagation() was called for this event.
originalMousePosition	The original mouse position of the IMGUI event, before it is transformed to the current target local coordinates.
pooled	Whether the event is allocated from a pool of events.
propagationPhase	The current propagation phase.
target	The target visual element that received this event. Unlike currentTarget, this target does not change when the event is sent to other elements along the propagation path.
timestamp	The time when the event was created.
tricklesDown	Whether this event is sent down the event propagation path during the

	TrickleDown phase.
actionKey	Returns true if the platform specific action key is pressed. This key is Command on macOS and Control otherwise.
altKey	Returns true if the Alt key is pressed.
button	Integer representing the pressed mouse button: 0 is left, 1 is right, 2 is center.
clickCount	Number of clicks.
commandKey	Returns true if the Windows/Command key is pressed.
ctrlKey	Returns true if the Control key is pressed.
currentTarget	The current target of the event. The current target is the element in the propagation path for which event handlers are currently being executed.
localMousePosition	The mouse position in the current target coordinate system.
modifiers	Flags holding the pressed modifier keys (Alt, Control, Shift, Windows/Command).
mouseDelta	The difference of the mouse position between the previous mouse event and the current mouse event.
mousePosition	The mouse position in the screen coordinate system.
shiftKey	Returns true if the Shift key is pressed.

Public Methods

Dispose	Implementation of IDisposable.
GetEventTypeId	Retrieves the type id for this event instance.
PreventDefault	Whether the default actions are prevented from being executed for this event.
StopImmediatePropagation	Immediately stops the propagation of the event. The event is not sent to other elements along the propagation path. This method prevents other event handlers from executing on the current target.
StopPropagation	Stops propagating this event. The event is not sent to other elements along the propagation path. This method does not prevent other event handlers from executing on the current target.
Dispose	Implementation of IDispose.
GetEventTypeId	Retrieves the type id for this event instance.

Protected Methods

Init	Resets all event members to their initial values.
Init	Resets all event members to their initial values.
Init	Resets the event members to their initial values.

Static Methods

RegisterEventType	Registers an event class to the event type system.
GetPooled	Gets an event from the event pool. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().
TypeId	Gets the type id for the event class.
GetPooled	Gets an event from the event pool and initializes it with the given values. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().

Description

Use this class to display a contextual menu.

Public Methods

DisplayMenu	Displays the contextual menu.
DisplayMenuIfEventMatches	Checks if the event triggers the display of the contextual menu. This method also displays the menu.

Description

Manipulator that displays a contextual menu when the user clicks the right mouse button or presses the menu key on the keyboard.

Constructors

ContextualMenuManipulator	Constructor.

Protected Methods

RegisterCallbacksOnTarget	Register the event callbacks on the manipulator target.
UnregisterCallbacksFromTarget	Unregister the event callbacks from the manipulator target.

Inherited Members

Properties

target	The element that handles the interaction.

Description

The event sent when a contextual menu requires menu items.

Properties

menu	The menu to populate.
triggerEvent	The event that triggered the ContextualMenuPopulateEvent.

Constructors

ContextualMenuPopulateEvent	Constructor.

Protected Methods

Init	Resets the event members to their initial values.

Static Methods

GetPooled	Retrieves an event from the event pool. Use this method to retrieve a mouse event and initialize the event, instead of creating a new mouse event. Events obtained from this method should be released back to the pool using Dispose().

Inherited Members

Properties

bubbles	Whether this event type bubbles up in the event propagation path.
currentTarget	The current target of the event. The current target is the element in the propagation path for which event handlers are currently being executed.
dispatch	Whether the event is being dispatched to a visual element. An event cannot be redispatched while it being dispatched. If you need to recursively dispatch an event, it is recommended that you use a copy of the event.
flags	Flags for the event.
imguiEvent	The IMGUIEvent at the source of this event. The source can be null since not all events are generated by IMGUI.

isDefaultPrevented	Return true if the default actions should not be executed for this event.
isImmediatePropagationStopped	Whether StopImmediatePropagation() was called for this event.
isPropagationStopped	Whether StopPropagation() was called for this event.
originalMousePosition	The original mouse position of the IMGUI event, before it is transformed to the current target local coordinates.
pooled	Whether the event is allocated from a pool of events.
propagationPhase	The current propagation phase.
target	The target visual element that received this event. Unlike currentTarget, this target does not change when the event is sent to other elements along the propagation path.
timestamp	The time when the event was created.
tricklesDown	Whether this event is sent down the event propagation path during the TrickleDown phase.
actionKey	Returns true if the platform specific action key is pressed. This key is Command on macOS and Control otherwise.
altKey	Returns true if the Alt key is pressed.
button	Integer representing the pressed mouse button: 0 is left, 1 is right, 2 is center.
clickCount	Number of clicks.
commandKey	Returns true if the Windows/Command key is pressed.
ctrlKey	Returns true if the Control key is pressed.
currentTarget	The current target of the event. The current target is the element in the propagation path for which event handlers are currently being executed.
localMousePosition	The mouse position in the current target coordinate system.
modifiers	Flags holding the pressed modifier keys (Alt, Control, Shift, Windows/Command).
mouseDelta	The difference of the mouse position between the previous mouse event and the current mouse event.
mousePosition	The mouse position in the screen coordinate system.
shiftKey	Returns true if the Shift key is pressed.

Public Methods

Dispose	Implementation of IDisposable.
GetEventTypeId	Retrieves the type id for this event instance.
PreventDefault	Whether the default actions are prevented from being executed for this event.
StopImmediatePropagation	Immediately stops the propagation of the event. The event is not sent to other elements along the propagation path. This method prevents other event handlers from executing on the current target.
StopPropagation	Stops propagating this event. The event is not sent to other elements along the propagation path. This method does not prevent other event handlers from executing on the current target.
Dispose	Implementation of IDispose.
GetEventTypeId	Retrieves the type id for this event instance.

Static Methods

RegisterEventType	Registers an event class to the event type system.
GetPooled	Gets an event from the event pool. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().
TypeId	Gets the type id for the event class.
GetPooled	Gets an event from the event pool and initializes it with the given values. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().

Description

This class is used during UXML template instantiation.

Description

Script interface for VisualElement cursor style property IStyle.cursor.

Properties

hotspot	The offset from the top left of the texture to use as the target point (must be within the bounds of the cursor).
texture	The texture to use for the cursor style. To use a texture as a cursor, import the texture with "Read/Write enabled" in the texture importer (or using the "Cursor" defaults).

Description

Event sent just before an element is detach from its parent, if the parent is the descendant of a panel.

Constructors

DetachFromPanelEvent	Constructor. Avoid newing events. Instead, use GetPooled() to get an event from a pool of reusable events.

Inherited Members

Properties

bubbles	Whether this event type bubbles up in the event propagation path.
currentTarget	The current target of the event. The current target is the element in the propagation path for which event handlers are currently being executed.
dispatch	Whether the event is being dispatched to a visual element. An event cannot be redispatched while it being dispatched. If you need to recursively dispatch an event, it is recommended that you use a copy of the event.
flags	Flags for the event.
imguiEvent	The IMGUIEvent at the source of this event. The source can be null since not all events are generated by IMGUI.
isDefaultPrevented	Return true if the default actions should not be executed for this event.
isImmediatePropagationStopped	Whether StopImmediatePropagation() was called for this event.
isPropagationStopped	Whether StopPropagation() was called for this event.
originalMousePosition	The original mouse position of the IMGUI event, before it is transformed to the current target local coordinates.
pooled	Whether the event is allocated from a pool of events.
propagationPhase	The current propagation phase.
target	The target visual element that received this event. Unlike currentTarget, this target does not change when the event is sent to other elements along the propagation path.
timestamp	The time when the event was created.

tricklesDown	Whether this event is sent down the event propagation path during the TrickleDown phase.
destinationPanel	In the case of AttachToPanelEvent, the panel to which the event target element is now attached. In the case of DetachFromPanelEvent, the panel to which the event target element will be attached.
originPanel	In the case of AttachToPanelEvent, the panel to which the event target element was attached. In the case of DetachFromPanelEvent, the panel from which the event target element is detached.

Public Methods

Dispose	Implementation of IDisposable.
GetEventTypeId	Retrieves the type id for this event instance.
PreventDefault	Whether the default actions are prevented from being executed for this event.
StopImmediatePropagation	Immediately stops the propagation of the event. The event is not sent to other elements along the propagation path. This method prevents other event handlers from executing on the current target.
StopPropagation	Stops propagating this event. The event is not sent to other elements along the propagation path. This method does not prevent other event handlers from executing on the current target.
Dispose	Implementation of IDispose.
GetEventTypeId	Retrieves the type id for this event instance.

Protected Methods

Init	Resets all event members to their initial values.
Init	Resets all event members to their initial values.
Init	Resets the event members to their initial values.

Static Methods

RegisterEventType	Registers an event class to the event type system.
GetPooled	Gets an event from the event pool. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().
TypeId	Gets the type id for the event class.
GetPooled	Gets an event from the event pool and initializes it with the given values. Use this

function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().

Implements interfaces:IDragAndDropEvent

Description

Base class for drag and drop events.

Inherited Members

Properties

bubbles	Whether this event type bubbles up in the event propagation path.
currentTarget	The current target of the event. The current target is the element in the propagation path for which event handlers are currently being executed.
dispatch	Whether the event is being dispatched to a visual element. An event cannot be redispatched while it being dispatched. If you need to recursively dispatch an event, it is recommended that you use a copy of the event.
flags	Flags for the event.
imguiEvent	The IMGUIEvent at the source of this event. The source can be null since not all events are generated by IMGUI.
isDefaultPrevented	Return true if the default actions should not be executed for this event.
isImmediatePropagationStopped	Whether StopImmediatePropagation() was called for this event.
isPropagationStopped	Whether StopPropagation() was called for this event.
originalMousePosition	The original mouse position of the IMGUI event, before it is transformed to the current target local coordinates.
pooled	Whether the event is allocated from a pool of events.
propagationPhase	The current propagation phase.
target	The target visual element that received this event. Unlike currentTarget, this target does not change when the event is sent to other elements along the propagation path.
timestamp	The time when the event was created.
tricklesDown	Whether this event is sent down the event propagation path during the TrickleDown phase.
actionKey	Returns true if the platform specific action key is pressed. This key is Command on macOS and Control otherwise.

altKey	Returns true if the Alt key is pressed.
button	Integer representing the pressed mouse button: 0 is left, 1 is right, 2 is center.
clickCount	Number of clicks.
commandKey	Returns true if the Windows/Command key is pressed.
ctrlKey	Returns true if the Control key is pressed.
currentTarget	The current target of the event. The current target is the element in the propagation path for which event handlers are currently being executed.
localMousePosition	The mouse position in the current target coordinate system.
modifiers	Flags holding the pressed modifier keys (Alt, Control, Shift, Windows/Command).
mouseDelta	The difference of the mouse position between the previous mouse event and the current mouse event.
mousePosition	The mouse position in the screen coordinate system.
shiftKey	Returns true if the Shift key is pressed.

Public Methods

Dispose	Implementation of IDisposable.
GetEventTypeId	Retrieves the type id for this event instance.
PreventDefault	Whether the default actions are prevented from being executed for this event.
StopImmediatePropagation	Immediately stops the propagation of the event. The event is not sent to other elements along the propagation path. This method prevents other event handlers from executing on the current target.
StopPropagation	Stops propagating this event. The event is not sent to other elements along the propagation path. This method does not prevent other event handlers from executing on the current target.
Dispose	Implementation of IDispose.
GetEventTypeId	Retrieves the type id for this event instance.

Protected Methods

Init	Resets all event members to their initial values.
Init	Resets all event members to their initial values.

Init	Resets the event members to their initial values.

Static Methods

RegisterEventType	Registers an event class to the event type system.
GetPooled	Gets an event from the event pool. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().
TypeId	Gets the type id for the event class.
GetPooled	Gets an event from the event pool and initializes it with the given values. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().

Description

Use the DragEnterEvent class to manage events that occur when dragging enters an element or one of its descendants. The DragEnterEvent is cancellable, it does not trickle down, and it does not bubble up.

Constructors

DragEnterEvent	Constructor. Avoid renewing events. Instead, use GetPooled() to get an event from a pool of reusable events.

Protected Methods

Init	Resets the event members to their initial values.

Inherited Members

Properties

bubbles	Whether this event type bubbles up in the event propagation path.
currentTarget	The current target of the event. The current target is the element in the propagation path for which event handlers are currently being executed.
dispatch	Whether the event is being dispatched to a visual element. An event cannot be redispatched while it being dispatched. If you need to recursively dispatch an event, it is recommended that you use a copy of the event.
flags	Flags for the event.
imguiEvent	The IMGUIEvent at the source of this event. The source can be null since not all events are generated by IMGUI.
isDefaultPrevented	Return true if the default actions should not be executed for this event.
isImmediatePropagationStopped	Whether StopImmediatePropagation() was called for this event.
isPropagationStopped	Whether StopPropagation() was called for this event.
originalMousePosition	The original mouse position of the IMGUI event, before it is transformed to the current target local coordinates.
pooled	Whether the event is allocated from a pool of events.
propagationPhase	The current propagation phase.

target	The target visual element that received this event. Unlike currentTarget, this target does not change when the event is sent to other elements along the propagation path.
timestamp	The time when the event was created.
tricklesDown	Whether this event is sent down the event propagation path during the TrickleDown phase.
actionKey	Returns true if the platform specific action key is pressed. This key is Command on macOS and Control otherwise.
altKey	Returns true if the Alt key is pressed.
button	Integer representing the pressed mouse button: 0 is left, 1 is right, 2 is center.
clickCount	Number of clicks.
commandKey	Returns true if the Windows/Command key is pressed.
ctrlKey	Returns true if the Control key is pressed.
currentTarget	The current target of the event. The current target is the element in the propagation path for which event handlers are currently being executed.
localMousePosition	The mouse position in the current target coordinate system.
modifiers	Flags holding the pressed modifier keys (Alt, Control, Shift, Windows/Command).
mouseDelta	The difference of the mouse position between the previous mouse event and the current mouse event.
mousePosition	The mouse position in the screen coordinate system.
shiftKey	Returns true if the Shift key is pressed.

Public Methods

Dispose	Implementation of IDisposable.
GetEventTypeId	Retrieves the type id for this event instance.
PreventDefault	Whether the default actions are prevented from being executed for this event.
StopImmediatePropagation	Immediately stops the propagation of the event. The event is not sent to other elements along the propagation path. This method prevents other event handlers from executing on the current target.
StopPropagation	Stops propagating this event. The event is not sent to other elements along the propagation path. This method does not prevent other event handlers from

	executing on the current target.
Dispose	Implementation of IDispose.
GetEventTypeId	Retrieves the type id for this event instance.

Static Methods

RegisterEventType	Registers an event class to the event type system.
GetPooled	Gets an event from the event pool. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().
TypeId	Gets the type id for the event class.
GetPooled	Gets an event from the event pool and initializes it with the given values. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().

Description

The event sent to a dragged element when the drag and drop process ends.

Constructors

DragExitedEvent	Constructor.

Protected Methods

Init	Resets the event members to their initial values.

Inherited Members

Properties

bubbles	Whether this event type bubbles up in the event propagation path.
currentTarget	The current target of the event. The current target is the element in the propagation path for which event handlers are currently being executed.
dispatch	Whether the event is being dispatched to a visual element. An event cannot be redispatched while it being dispatched. If you need to recursively dispatch an event, it is recommended that you use a copy of the event.
flags	Flags for the event.
imguiEvent	The IMGUIEvent at the source of this event. The source can be null since not all events are generated by IMGUI.
isDefaultPrevented	Return true if the default actions should not be executed for this event.
isImmediatePropagationStopped	Whether StopImmediatePropagation() was called for this event.
isPropagationStopped	Whether StopPropagation() was called for this event.
originalMousePosition	The original mouse position of the IMGUI event, before it is transformed to the current target local coordinates.
pooled	Whether the event is allocated from a pool of events.
propagationPhase	The current propagation phase.
target	The target visual element that received this event. Unlike currentTarget, this target does not change when the event is sent to other elements

	along the propagation path.
timestamp	The time when the event was created.
tricklesDown	Whether this event is sent down the event propagation path during the TrickleDown phase.
actionKey	Returns true if the platform specific action key is pressed. This key is Command on macOS and Control otherwise.
altKey	Returns true if the Alt key is pressed.
button	Integer representing the pressed mouse button: 0 is left, 1 is right, 2 is center.
clickCount	Number of clicks.
commandKey	Returns true if the Windows/Command key is pressed.
ctrlKey	Returns true if the Control key is pressed.
currentTarget	The current target of the event. The current target is the element in the propagation path for which event handlers are currently being executed.
localMousePosition	The mouse position in the current target coordinate system.
modifiers	Flags holding the pressed modifier keys (Alt, Control, Shift, Windows/Command).
mouseDelta	The difference of the mouse position between the previous mouse event and the current mouse event.
mousePosition	The mouse position in the screen coordinate system.
shiftKey	Returns true if the Shift key is pressed.

Public Methods

Dispose	Implementation of IDisposable.
GetEventTypeId	Retrieves the type id for this event instance.
PreventDefault	Whether the default actions are prevented from being executed for this event.
StopImmediatePropagation	Immediately stops the propagation of the event. The event is not sent to other elements along the propagation path. This method prevents other event handlers from executing on the current target.
StopPropagation	Stops propagating this event. The event is not sent to other elements along the propagation path. This method does not prevent other event handlers from executing on the current target.

Dispose	Implementation of IDispose.
GetEventTypeId	Retrieves the type id for this event instance.

Static Methods

RegisterEventType	Registers an event class to the event type system.
GetPooled	Gets an event from the event pool. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().
TypeId	Gets the type id for the event class.
GetPooled	Gets an event from the event pool and initializes it with the given values. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().

Description

Use the DragLeaveEvent class to manage events sent when dragging leaves an element or one of its descendants. The DragLeaveEvent is cancellable, it does not trickle down, and it does not bubble up.

Constructors

DragLeaveEvent	Constructor. Avoid renewing events. Instead, use GetPooled() to get an event from a pool of reusable events.

Protected Methods

Init	Resets the event members to their initial values.

Inherited Members

Properties

bubbles	Whether this event type bubbles up in the event propagation path.
currentTarget	The current target of the event. The current target is the element in the propagation path for which event handlers are currently being executed.
dispatch	Whether the event is being dispatched to a visual element. An event cannot be redispatched while it being dispatched. If you need to recursively dispatch an event, it is recommended that you use a copy of the event.
flags	Flags for the event.
imguiEvent	The IMGUIEvent at the source of this event. The source can be null since not all events are generated by IMGUI.
isDefaultPrevented	Return true if the default actions should not be executed for this event.
isImmediatePropagationStopped	Whether StopImmediatePropagation() was called for this event.
isPropagationStopped	Whether StopPropagation() was called for this event.
originalMousePosition	The original mouse position of the IMGUI event, before it is transformed to the current target local coordinates.
pooled	Whether the event is allocated from a pool of events.
propagationPhase	The current propagation phase.

target	The target visual element that received this event. Unlike currentTarget, this target does not change when the event is sent to other elements along the propagation path.
timestamp	The time when the event was created.
tricklesDown	Whether this event is sent down the event propagation path during the TrickleDown phase.
actionKey	Returns true if the platform specific action key is pressed. This key is Command on macOS and Control otherwise.
altKey	Returns true if the Alt key is pressed.
button	Integer representing the pressed mouse button: 0 is left, 1 is right, 2 is center.
clickCount	Number of clicks.
commandKey	Returns true if the Windows/Command key is pressed.
ctrlKey	Returns true if the Control key is pressed.
currentTarget	The current target of the event. The current target is the element in the propagation path for which event handlers are currently being executed.
localMousePosition	The mouse position in the current target coordinate system.
modifiers	Flags holding the pressed modifier keys (Alt, Control, Shift, Windows/Command).
mouseDelta	The difference of the mouse position between the previous mouse event and the current mouse event.
mousePosition	The mouse position in the screen coordinate system.
shiftKey	Returns true if the Shift key is pressed.

Public Methods

Dispose	Implementation of IDisposable.
GetEventTypeId	Retrieves the type id for this event instance.
PreventDefault	Whether the default actions are prevented from being executed for this event.
StopImmediatePropagation	Immediately stops the propagation of the event. The event is not sent to other elements along the propagation path. This method prevents other event handlers from executing on the current target.
StopPropagation	Stops propagating this event. The event is not sent to other elements along the propagation path. This method does not prevent other event handlers from

	executing on the current target.
Dispose	Implementation of IDispose.
GetEventTypeId	Retrieves the type id for this event instance.

Static Methods

RegisterEventType	Registers an event class to the event type system.
GetPooled	Gets an event from the event pool. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().
TypeId	Gets the type id for the event class.
GetPooled	Gets an event from the event pool and initializes it with the given values. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().

Description

The event sent to an element when another element is dragged and dropped on the element.

Constructors

DragPerformEvent	Constructor.

Inherited Members

Properties

bubbles	Whether this event type bubbles up in the event propagation path.
currentTarget	The current target of the event. The current target is the element in the propagation path for which event handlers are currently being executed.
dispatch	Whether the event is being dispatched to a visual element. An event cannot be redispatched while it being dispatched. If you need to recursively dispatch an event, it is recommended that you use a copy of the event.
flags	Flags for the event.
imguiEvent	The IMGUIEvent at the source of this event. The source can be null since not all events are generated by IMGUI.
isDefaultPrevented	Return true if the default actions should not be executed for this event.
isImmediatePropagationStopped	Whether StopImmediatePropagation() was called for this event.
isPropagationStopped	Whether StopPropagation() was called for this event.
originalMousePosition	The original mouse position of the IMGUI event, before it is transformed to the current target local coordinates.
pooled	Whether the event is allocated from a pool of events.
propagationPhase	The current propagation phase.
target	The target visual element that received this event. Unlike currentTarget, this target does not change when the event is sent to other elements along the propagation path.
timestamp	The time when the event was created.
tricklesDown	Whether this event is sent down the event propagation path during the

	TrickleDown phase.
actionKey	Returns true if the platform specific action key is pressed. This key is Command on macOS and Control otherwise.
altKey	Returns true if the Alt key is pressed.
button	Integer representing the pressed mouse button: 0 is left, 1 is right, 2 is center.
clickCount	Number of clicks.
commandKey	Returns true if the Windows/Command key is pressed.
ctrlKey	Returns true if the Control key is pressed.
currentTarget	The current target of the event. The current target is the element in the propagation path for which event handlers are currently being executed.
localMousePosition	The mouse position in the current target coordinate system.
modifiers	Flags holding the pressed modifier keys (Alt, Control, Shift, Windows/Command).
mouseDelta	The difference of the mouse position between the previous mouse event and the current mouse event.
mousePosition	The mouse position in the screen coordinate system.
shiftKey	Returns true if the Shift key is pressed.

Public Methods

Dispose	Implementation of IDisposable.
GetEventTypeId	Retrieves the type id for this event instance.
PreventDefault	Whether the default actions are prevented from being executed for this event.
StopImmediatePropagation	Immediately stops the propagation of the event. The event is not sent to other elements along the propagation path. This method prevents other event handlers from executing on the current target.
StopPropagation	Stops propagating this event. The event is not sent to other elements along the propagation path. This method does not prevent other event handlers from executing on the current target.
Dispose	Implementation of IDispose.
GetEventTypeId	Retrieves the type id for this event instance.

Protected Methods

Init	Resets all event members to their initial values.
Init	Resets all event members to their initial values.
Init	Resets the event members to their initial values.

Static Methods

RegisterEventType	Registers an event class to the event type system.
GetPooled	Gets an event from the event pool. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().
TypeId	Gets the type id for the event class.
GetPooled	Gets an event from the event pool and initializes it with the given values. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().

Description

The event sent when the element being dragged enters a possible drop target.

Constructors

DragUpdatedEvent	Constructor.

Inherited Members

Properties

bubbles	Whether this event type bubbles up in the event propagation path.
currentTarget	The current target of the event. The current target is the element in the propagation path for which event handlers are currently being executed.
dispatch	Whether the event is being dispatched to a visual element. An event cannot be redispatched while it being dispatched. If you need to recursively dispatch an event, it is recommended that you use a copy of the event.
flags	Flags for the event.
imguiEvent	The IMGUIEvent at the source of this event. The source can be null since not all events are generated by IMGUI.
isDefaultPrevented	Return true if the default actions should not be executed for this event.
isImmediatePropagationStopped	Whether StopImmediatePropagation() was called for this event.
isPropagationStopped	Whether StopPropagation() was called for this event.
originalMousePosition	The original mouse position of the IMGUI event, before it is transformed to the current target local coordinates.
pooled	Whether the event is allocated from a pool of events.
propagationPhase	The current propagation phase.
target	The target visual element that received this event. Unlike currentTarget, this target does not change when the event is sent to other elements along the propagation path.
timestamp	The time when the event was created.
tricklesDown	Whether this event is sent down the event propagation path during the

	TrickleDown phase.
actionKey	Returns true if the platform specific action key is pressed. This key is Command on macOS and Control otherwise.
altKey	Returns true if the Alt key is pressed.
button	Integer representing the pressed mouse button: 0 is left, 1 is right, 2 is center.
clickCount	Number of clicks.
commandKey	Returns true if the Windows/Command key is pressed.
ctrlKey	Returns true if the Control key is pressed.
currentTarget	The current target of the event. The current target is the element in the propagation path for which event handlers are currently being executed.
localMousePosition	The mouse position in the current target coordinate system.
modifiers	Flags holding the pressed modifier keys (Alt, Control, Shift, Windows/Command).
mouseDelta	The difference of the mouse position between the previous mouse event and the current mouse event.
mousePosition	The mouse position in the screen coordinate system.
shiftKey	Returns true if the Shift key is pressed.

Public Methods

Dispose	Implementation of IDisposable.
GetEventTypeId	Retrieves the type id for this event instance.
PreventDefault	Whether the default actions are prevented from being executed for this event.
StopImmediatePropagation	Immediately stops the propagation of the event. The event is not sent to other elements along the propagation path. This method prevents other event handlers from executing on the current target.
StopPropagation	Stops propagating this event. The event is not sent to other elements along the propagation path. This method does not prevent other event handlers from executing on the current target.
Dispose	Implementation of IDispose.
GetEventTypeId	Retrieves the type id for this event instance.

Protected Methods

Init	Resets all event members to their initial values.
Init	Resets all event members to their initial values.
Init	Resets the event members to their initial values.

Static Methods

RegisterEventType	Registers an event class to the event type system.
GetPooled	Gets an event from the event pool. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().
TypeId	Gets the type id for the event class.
GetPooled	Gets an event from the event pool and initializes it with the given values. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().

Description

The base class for all UIElements events.

Properties

bubbles	Whether this event type bubbles up in the event propagation path.
currentTarget	The current target of the event. The current target is the element in the propagation path for which event handlers are currently being executed.
dispatch	Whether the event is being dispatched to a visual element. An event cannot be redispatched while it being dispatched. If you need to recursively dispatch an event, it is recommended that you use a copy of the event.
flags	Flags for the event.
imguiEvent	The IMGUIEvent at the source of this event. The source can be null since not all events are generated by IMGUI.
isDefaultPrevented	Return true if the default actions should not be executed for this event.
isImmediatePropagationStopped	Whether StopImmediatePropagation() was called for this event.
isPropagationStopped	Whether StopPropagation() was called for this event.
originalMousePosition	The original mouse position of the IMGUI event, before it is transformed to the current target local coordinates.
pooled	Whether the event is allocated from a pool of events.
propagationPhase	The current propagation phase.
target	The target visual element that received this event. Unlike currentTarget, this target does not change when the event is sent to other elements along the propagation path.
timestamp	The time when the event was created.
tricklesDown	Whether this event is sent down the event propagation path during the TrickleDown phase.

Public Methods

Dispose	Implementation of IDisposable.

GetEventTypeId	Retrieves the type id for this event instance.
PreventDefault	Whether the default actions are prevented from being executed for this event.
StopImmediatePropagation	Immediately stops the propagation of the event. The event is not sent to other elements along the propagation path. This method prevents other event handlers from executing on the current target.
StopPropagation	Stops propagating this event. The event is not sent to other elements along the propagation path. This method does not prevent other event handlers from executing on the current target.

Protected Methods

Init	Resets all event members to their initial values.

Static Methods

RegisterEventType	Registers an event class to the event type system.

Description

Generic base class for events, implementing event pooling and automatic registration to the event type system.

Public Methods

Dispose	Implementation of IDispose.
GetEventTypeId	Retrieves the type id for this event instance.

Protected Methods

Init	Resets all event members to their initial values.

Static Methods

GetPooled	Gets an event from the event pool. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().
TypeId	Gets the type id for the event class.

Inherited Members

Properties

bubbles	Whether this event type bubbles up in the event propagation path.
currentTarget	The current target of the event. The current target is the element in the propagation path for which event handlers are currently being executed.
dispatch	Whether the event is being dispatched to a visual element. An event cannot be redispatched while it being dispatched. If you need to recursively dispatch an event, it is recommended that you use a copy of the event.
flags	Flags for the event.
imguiEvent	The IMGUIEvent at the source of this event. The source can be null since not all events are generated by IMGUI.
isDefaultPrevented	Return true if the default actions should not be executed for this event.
isImmediatePropagationStopped	Whether StopImmediatePropagation() was called for this event.

isPropagationStopped	Whether StopPropagation() was called for this event.
originalMousePosition	The original mouse position of the IMGUI event, before it is transformed to the current target local coordinates.
pooled	Whether the event is allocated from a pool of events.
propagationPhase	The current propagation phase.
target	The target visual element that received this event. Unlike currentTarget, this target does not change when the event is sent to other elements along the propagation path.
timestamp	The time when the event was created.
tricklesDown	Whether this event is sent down the event propagation path during the TrickleDown phase.

Public Methods

PreventDefault	Whether the default actions are prevented from being executed for this event.
StopImmediatePropagation	Immediately stops the propagation of the event. The event is not sent to other elements along the propagation path. This method prevents other event handlers from executing on the current target.
StopPropagation	Stops propagating this event. The event is not sent to other elements along the propagation path. This method does not prevent other event handlers from executing on the current target.

Static Methods

RegisterEventType	Registers an event class to the event type system.

Description

The event sent when an element should execute a command.

Constructors

ExecuteCommandEvent	Constructor.

Inherited Members

Properties

commandName	Name of the command.
bubbles	Whether this event type bubbles up in the event propagation path.
currentTarget	The current target of the event. The current target is the element in the propagation path for which event handlers are currently being executed.
dispatch	Whether the event is being dispatched to a visual element. An event cannot be redispatched while it being dispatched. If you need to recursively dispatch an event, it is recommended that you use a copy of the event.
flags	Flags for the event.
imguiEvent	The IMGUIEvent at the source of this event. The source can be null since not all events are generated by IMGUI.
isDefaultPrevented	Return true if the default actions should not be executed for this event.
isImmediatePropagationStopped	Whether StopImmediatePropagation() was called for this event.
isPropagationStopped	Whether StopPropagation() was called for this event.
originalMousePosition	The original mouse position of the IMGUI event, before it is transformed to the current target local coordinates.
pooled	Whether the event is allocated from a pool of events.
propagationPhase	The current propagation phase.
target	The target visual element that received this event. Unlike currentTarget, this target does not change when the event is sent to other elements along the propagation path.
timestamp	The time when the event was created.

tricklesDown	Whether this event is sent down the event propagation path during the TrickleDown phase.

Public Methods

Dispose	Implementation of IDisposable.
GetEventTypeId	Retrieves the type id for this event instance.
PreventDefault	Whether the default actions are prevented from being executed for this event.
StopImmediatePropagation	Immediately stops the propagation of the event. The event is not sent to other elements along the propagation path. This method prevents other event handlers from executing on the current target.
StopPropagation	Stops propagating this event. The event is not sent to other elements along the propagation path. This method does not prevent other event handlers from executing on the current target.
Dispose	Implementation of IDispose.
GetEventTypeId	Retrieves the type id for this event instance.

Protected Methods

Init	Resets the event members to their initial values.
Init	Resets all event members to their initial values.
Init	Resets all event members to their initial values.

Static Methods

GetPooled	Gets an event from the event pool and initializes it with the given values. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().
RegisterEventType	Registers an event class to the event type system.
GetPooled	Gets an event from the event pool. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().
TypeId	Gets the type id for the event class.

Description

Base class for objects that can get the focus.

The focus is used to designate an element that will receive keyboard events.

Properties

canGrabFocus	Return true if the element can be focused.
focusController	Return the focus controller for this element.
focusIndex	An integer used to sort focusables in the focus ring. A negative value means that the element can not be focused.

Public Methods

Blur	Tell the element to release the focus.
Focus	Attempt to give the focus to this element.

Inherited Members

Public Methods

HandleEvent	Handle an event, most often by executing the callbacks associated with the event.
HasBubbleUpHandlers	Return true if event handlers for the event propagation BubbleUp phase have been attached on this object.
HasTrickleDownHandlers	Returns true if event handlers, for the event propagation TrickleDown phase, are attached to this object.
RegisterCallback	Adds an event handler to the instance. If the event handler has already been registered for the same phase (either TrickleDown or BubbleUp) then this method has no effect.
SendEvent	Sends an event to the event handler.
UnregisterCallback	Remove callback from the instance.

Description

Base class for defining in which direction the focus moves in a focus ring.

Focus ring implementations can move the focus in various direction; they can derive from this class to formalize the various ways the focus can change from one element to the other.

Static Properties

lastValue	Last value for the direction defined by this class.
none	The null direction. This is usually used when the focus stays on the same element.
unspecified	Focus came from an unspecified direction, for example after a mouse down.

Operators

int	The underlying integer value for this direction.

Description

Class in charge of managing the focus inside a Panel.

Each Panel should have an instance of this class. The instance holds the currently focused element and is responsible for changing it.

Properties

focusedElement	The currently focused element.

Constructors

FocusController	Constructor.

Public Methods

SwitchFocusOnEvent	Ask the controller to change the focus according to the event. The focus controller will use its focus ring to choose the next element to be focused.

Description

Event sent immediately after an element has gained focus. This event trickles down, it does not bubble up, and it cannot be cancelled.

Constructors

FocusEvent	Constructor. Avoid newing events. Instead, use GetPooled() to get an event from a pool of reusable events.

Inherited Members

Properties

bubbles	Whether this event type bubbles up in the event propagation path.
currentTarget	The current target of the event. The current target is the element in the propagation path for which event handlers are currently being executed.
dispatch	Whether the event is being dispatched to a visual element. An event cannot be redispatched while it being dispatched. If you need to recursively dispatch an event, it is recommended that you use a copy of the event.
flags	Flags for the event.
imguiEvent	The IMGUIEvent at the source of this event. The source can be null since not all events are generated by IMGUI.
isDefaultPrevented	Return true if the default actions should not be executed for this event.
isImmediatePropagationStopped	Whether StopImmediatePropagation() was called for this event.
isPropagationStopped	Whether StopPropagation() was called for this event.
originalMousePosition	The original mouse position of the IMGUI event, before it is transformed to the current target local coordinates.
pooled	Whether the event is allocated from a pool of events.
propagationPhase	The current propagation phase.
target	The target visual element that received this event. Unlike currentTarget, this target does not change when the event is sent to other elements along the propagation path.
timestamp	The time when the event was created.

tricklesDown	Whether this event is sent down the event propagation path during the TrickleDown phase.
direction	Direction of the focus change.
relatedTarget	For FocusOut and Blur events, contains the element that gains the focus. For FocusIn and Focus events, contains the element that loses the focus.

Public Methods

Dispose	Implementation of IDisposable.
GetEventTypeId	Retrieves the type id for this event instance.
PreventDefault	Whether the default actions are prevented from being executed for this event.
StopImmediatePropagation	Immediately stops the propagation of the event. The event is not sent to other elements along the propagation path. This method prevents other event handlers from executing on the current target.
StopPropagation	Stops propagating this event. The event is not sent to other elements along the propagation path. This method does not prevent other event handlers from executing on the current target.
Dispose	Implementation of IDispose.
GetEventTypeId	Retrieves the type id for this event instance.

Protected Methods

Init	Resets all event members to their initial values.
Init	Resets all event members to their initial values.
Init	Resets the event members to their initial values.

Static Methods

RegisterEventType	Registers an event class to the event type system.
GetPooled	Gets an event from the event pool. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().
TypeId	Gets the type id for the event class.
GetPooled	Gets an event from the event pool and initializes the event with the given values. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().

Implements interfaces:IFocusEvent

Description

Base class for focus related events.

Properties

direction	Direction of the focus change.
relatedTarget	For FocusOut and Blur events, contains the element that gains the focus. For FocusIn and Focus events, contains the element that loses the focus.

Protected Methods

Init	Resets the event members to their initial values.

Static Methods

GetPooled	Gets an event from the event pool and initializes the event with the given values. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().

Inherited Members

Properties

bubbles	Whether this event type bubbles up in the event propagation path.
currentTarget	The current target of the event. The current target is the element in the propagation path for which event handlers are currently being executed.
dispatch	Whether the event is being dispatched to a visual element. An event cannot be redispatched while it being dispatched. If you need to recursively dispatch an event, it is recommended that you use a copy of the event.
flags	Flags for the event.
imguiEvent	The IMGUIEvent at the source of this event. The source can be null since not all events are generated by IMGUI.
isDefaultPrevented	Return true if the default actions should not be executed for this event.

isImmediatePropagationStopped	Whether StopImmediatePropagation() was called for this event.
isPropagationStopped	Whether StopPropagation() was called for this event.
originalMousePosition	The original mouse position of the IMGUI event, before it is transformed to the current target local coordinates.
pooled	Whether the event is allocated from a pool of events.
propagationPhase	The current propagation phase.
target	The target visual element that received this event. Unlike currentTarget, this target does not change when the event is sent to other elements along the propagation path.
timestamp	The time when the event was created.
tricklesDown	Whether this event is sent down the event propagation path during the TrickleDown phase.

Public Methods

Dispose	Implementation of IDisposable.
GetEventTypeId	Retrieves the type id for this event instance.
PreventDefault	Whether the default actions are prevented from being executed for this event.
StopImmediatePropagation	Immediately stops the propagation of the event. The event is not sent to other elements along the propagation path. This method prevents other event handlers from executing on the current target.
StopPropagation	Stops propagating this event. The event is not sent to other elements along the propagation path. This method does not prevent other event handlers from executing on the current target.
Dispose	Implementation of IDispose.
GetEventTypeId	Retrieves the type id for this event instance.

Static Methods

RegisterEventType	Registers an event class to the event type system.
GetPooled	Gets an event from the event pool. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().
TypeId	Gets the type id for the event class.

Description

Event sent immediately before an element gains focus. This event trickles down and bubbles up. This event cannot be cancelled.

Constructors

FocusInEvent	Constructor.

Protected Methods

Init	Resets the event members to their initial values.

Inherited Members

Properties

bubbles	Whether this event type bubbles up in the event propagation path.
currentTarget	The current target of the event. The current target is the element in the propagation path for which event handlers are currently being executed.
dispatch	Whether the event is being dispatched to a visual element. An event cannot be redispatched while it being dispatched. If you need to recursively dispatch an event, it is recommended that you use a copy of the event.
flags	Flags for the event.
imguiEvent	The IMGUIEvent at the source of this event. The source can be null since not all events are generated by IMGUI.
isDefaultPrevented	Return true if the default actions should not be executed for this event.
isImmediatePropagationStopped	Whether StopImmediatePropagation() was called for this event.
isPropagationStopped	Whether StopPropagation() was called for this event.
originalMousePosition	The original mouse position of the IMGUI event, before it is transformed to the current target local coordinates.
pooled	Whether the event is allocated from a pool of events.
propagationPhase	The current propagation phase.
target	The target visual element that received this event. Unlike currentTarget,

	this target does not change when the event is sent to other elements along the propagation path.
timestamp	The time when the event was created.
tricklesDown	Whether this event is sent down the event propagation path during the TrickleDown phase.
direction	Direction of the focus change.
relatedTarget	For FocusOut and Blur events, contains the element that gains the focus. For FocusIn and Focus events, contains the element that loses the focus.

Public Methods

Dispose	Implementation of IDisposable.
GetEventTypeId	Retrieves the type id for this event instance.
PreventDefault	Whether the default actions are prevented from being executed for this event.
StopImmediatePropagation	Immediately stops the propagation of the event. The event is not sent to other elements along the propagation path. This method prevents other event handlers from executing on the current target.
StopPropagation	Stops propagating this event. The event is not sent to other elements along the propagation path. This method does not prevent other event handlers from executing on the current target.
Dispose	Implementation of IDispose.
GetEventTypeId	Retrieves the type id for this event instance.

Static Methods

RegisterEventType	Registers an event class to the event type system.
GetPooled	Gets an event from the event pool. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().
TypeId	Gets the type id for the event class.
GetPooled	Gets an event from the event pool and initializes the event with the given values. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().

Description

Event sent immediately before an element loses focus. This event trickles down and bubbles up. This event cannot be cancelled.

Constructors

FocusOutEvent	Constructor. Avoid newing events. Instead, use GetPooled() to get an event from a pool of reusable events.

Protected Methods

Init	Resets the event members to their initial values.

Inherited Members

Properties

bubbles	Whether this event type bubbles up in the event propagation path.
currentTarget	The current target of the event. The current target is the element in the propagation path for which event handlers are currently being executed.
dispatch	Whether the event is being dispatched to a visual element. An event cannot be redispatched while it being dispatched. If you need to recursively dispatch an event, it is recommended that you use a copy of the event.
flags	Flags for the event.
imguiEvent	The IMGUIEvent at the source of this event. The source can be null since not all events are generated by IMGUI.
isDefaultPrevented	Return true if the default actions should not be executed for this event.
isImmediatePropagationStopped	Whether StopImmediatePropagation() was called for this event.
isPropagationStopped	Whether StopPropagation() was called for this event.
originalMousePosition	The original mouse position of the IMGUI event, before it is transformed to the current target local coordinates.
pooled	Whether the event is allocated from a pool of events.
propagationPhase	The current propagation phase.

target	The target visual element that received this event. Unlike currentTarget, this target does not change when the event is sent to other elements along the propagation path.
timestamp	The time when the event was created.
tricklesDown	Whether this event is sent down the event propagation path during the TrickleDown phase.
direction	Direction of the focus change.
relatedTarget	For FocusOut and Blur events, contains the element that gains the focus. For FocusIn and Focus events, contains the element that loses the focus.

Public Methods

Dispose	Implementation of IDisposable.
GetEventTypeId	Retrieves the type id for this event instance.
PreventDefault	Whether the default actions are prevented from being executed for this event.
StopImmediatePropagation	Immediately stops the propagation of the event. The event is not sent to other elements along the propagation path. This method prevents other event handlers from executing on the current target.
StopPropagation	Stops propagating this event. The event is not sent to other elements along the propagation path. This method does not prevent other event handlers from executing on the current target.
Dispose	Implementation of IDispose.
GetEventTypeId	Retrieves the type id for this event instance.

Static Methods

RegisterEventType	Registers an event class to the event type system.
GetPooled	Gets an event from the event pool. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().
TypeId	Gets the type id for the event class.
GetPooled	Gets an event from the event pool and initializes the event with the given values. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().

Description

Event sent after layout calculations, when the position or the dimension of an element changes. This event cannot be cancelled, it does not trickle down, and it does not bubble up.

Properties

newRect	The new dimensions of the element.
oldRect	The old dimensions of the element.

Constructors

GeometryChangedEvent	Constructor.

Protected Methods

Init	Resets the event values to their initial values.

Static Methods

GetPooled	Gets an event from the event pool and initializes the event with the specified values. Use this method instead of instancing new events. Use Dispose() to release events back to the event pool.

Inherited Members

Properties

bubbles	Whether this event type bubbles up in the event propagation path.
currentTarget	The current target of the event. The current target is the element in the propagation path for which event handlers are currently being executed.
dispatch	Whether the event is being dispatched to a visual element. An event cannot be redispatched while it being dispatched. If you need to recursively dispatch an event, it is recommended that you use a copy of the event.
flags	Flags for the event.
imguiEvent	The IMGUIEvent at the source of this event. The source can be null since

	not all events are generated by IMGUI.
isDefaultPrevented	Return true if the default actions should not be executed for this event.
isImmediatePropagationStopped	Whether StopImmediatePropagation() was called for this event.
isPropagationStopped	Whether StopPropagation() was called for this event.
originalMousePosition	The original mouse position of the IMGUI event, before it is transformed to the current target local coordinates.
pooled	Whether the event is allocated from a pool of events.
propagationPhase	The current propagation phase.
target	The target visual element that received this event. Unlike currentTarget, this target does not change when the event is sent to other elements along the propagation path.
timestamp	The time when the event was created.
tricklesDown	Whether this event is sent down the event propagation path during the TrickleDown phase.

Public Methods

Dispose	Implementation of IDisposable.
GetEventTypeId	Retrieves the type id for this event instance.
PreventDefault	Whether the default actions are prevented from being executed for this event.
StopImmediatePropagation	Immediately stops the propagation of the event. The event is not sent to other elements along the propagation path. This method prevents other event handlers from executing on the current target.
StopPropagation	Stops propagating this event. The event is not sent to other elements along the propagation path. This method does not prevent other event handlers from executing on the current target.
Dispose	Implementation of IDispose.
GetEventTypeId	Retrieves the type id for this event instance.

Static Methods

RegisterEventType	Registers an event class to the event type system.
GetPooled	Gets an event from the event pool. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().

TypeId	Gets the type id for the event class.

Description

A VisualElement representing a source texture.

Properties

image	The source texture of the Image element.
sourceRect	The source rectangle inside the texture relative to the top left corner.
uv	The base texture coordinates of the Image relative to the bottom left corner.

Inherited Members

Static Properties

defaultFocusIndex	The default focus index for newly created elements.

Properties

canGrabFocus	Return true if the element can be focused.
focusController	Return the focus controller for this element.
focusIndex	An integer used to sort focusables in the focus ring. A negative value means that the element can not be focused.
childCount	Number of child elements in this object's contentContainer
clippingOptions	Should this element clip painting to its boundaries.
contentContainer	child elements are added to this element, usually this
dataWatch	Access to this element data watch interface.
enabledInHierarchy	Returns true if the VisualElement is enabled in its own hierarchy.
enabledSelf	Returns true if the VisualElement is enabled locally.
persistenceKey	Used for view data persistence (ie. tree expanded states, scroll position, zoom level).
schedule	Retrieves this VisualElement's IVisualElementScheduler
shadow	Access to this element physical hierarchy
style	Reference to the style object of this element.

this[int]	Access to this element physical hierarchy
tooltip	Text to display inside an information box after the user hovers the element for a small amount of time.
userData	This property can be used to associate application-specific user data with this VisualElement.

Public Methods

HandleEvent	Handle an event, most often by executing the callbacks associated with the event.
HasBubbleUpHandlers	Return true if event handlers for the event propagation BubbleUp phase have been attached on this object.
HasTrickleDownHandlers	Returns true if event handlers, for the event propagation TrickleDown phase, are attached to this object.
RegisterCallback	Adds an event handler to the instance. If the event handler has already been registered for the same phase (either TrickleDown or BubbleUp) then this method has no effect.
SendEvent	Sends an event to the event handler.
UnregisterCallback	Remove callback from the instance.
Blur	Tell the element to release the focus.
Focus	Attempt to give the focus to this element.
Add	Add an element to this element's contentContainer
AddStyleSheetPath	Adds this stylesheet file to this element list of applied styles
BringToFront	Brings this element to the end of its parent children list. The element will be visually in front of any overlapping sibling elements.
Children	Returns the elements from its contentContainer
Clear	Remove all child elements from this element's contentContainer
Contains	Returns true if the element is a direct child of this VisualElement
ElementAt	Retrieves the child element at position
EnableInClassList	Enables or disables the class with the given name.
FindAncestorUserData	Searchs up the hierachy of this VisualElement and retrieves stored userData, if any is found.
FindCommonAncestor	Finds the lowest commont ancestor between two VisualElements inside

	the VisualTree hierarchy
GetEnumerator	Allows to iterate into this elements children
GetFirstAncestorOfType	Walks up the hierarchy, starting from this element's parent, and returns the first VisualElement of this type
GetFirstOfType	Walks up the hierarchy, starting from this element, and returns the first VisualElement of this type
GetFullHierarchicalPersistenceKey	Combine this VisualElement's VisualElement.persistenceKey with those of its parents to create a more unique key for use with VisualElement.GetOrCreatePersistentData.
GetOrCreatePersistentData	Takes a reference to an existing persisted object and a key and returns the object either filled with the persisted state or as-is.
HasStyleSheetPath	Checks if this stylesheet file is in this element list of applied styles
IndexOf	Retrieves the child index of the specified VisualElement.
Insert	Insert an element into this element's contentContainer
MarkDirtyRepaint	Triggers a repaint of the VisualElement on the next frame.
OnPersistentDataReady	Called when the persistent data is accessible and/or when the data or persistence key have changed (VisualElement is properly parented).
OverwriteFromPersistedData	Overwrite object from the persistent data store.
PlaceBehind	Places this element right before the sibling element in their parent children list. If the element and the sibling position overlap, the element will be visually behind of its sibling.
PlaceInFront	Places this element right after the sibling element in their parent children list. If the element and the sibling position overlap, the element will be visually in front of its sibling.
Remove	Removes this child from the hierarchy
RemoveAt	Remove the child element located at this position from this element's contentContainer
RemoveFromHierarchy	Removes this element from its parent hierarchy
RemoveStyleSheetPath	Removes this stylesheet file from this element list of applied styles
SavePersistentData	Write persistence data to file.
SendEvent	Sends an event to the event handler.
SendToBack	Sends this element to the beginning of its parent children list. The element will be visually behind any overlapping sibling elements.

SetEnabled	Changes whether the current VisualElement is enabled or not. When disabled, a VisualElement does not receive most events.
Sort	Reorders child elements from this VisualElement contentContainer.
ToggleInClassList	Toggles between adding and removing the given class name from the class list.

Protected Methods

OnStyleResolved	Callback when the styles of an object have changed.

Description

Class used to send a IMGUI event that has no equivalent UIElements event.

Constructors

IMGUIEvent	Constructor. Use GetPooled() to get an event from a pool of reusable events.

Protected Methods

Init	Resets the event members to their initial values.

Static Methods

GetPooled	Gets an event from the event pool and initializes it with the given values. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().

Inherited Members

Properties

bubbles	Whether this event type bubbles up in the event propagation path.
currentTarget	The current target of the event. The current target is the element in the propagation path for which event handlers are currently being executed.
dispatch	Whether the event is being dispatched to a visual element. An event cannot be redispatched while it being dispatched. If you need to recursively dispatch an event, it is recommended that you use a copy of the event.
flags	Flags for the event.
imguiEvent	The IMGUIEvent at the source of this event. The source can be null since not all events are generated by IMGUI.
isDefaultPrevented	Return true if the default actions should not be executed for this event.
isImmediatePropagationStopped	Whether StopImmediatePropagation() was called for this event.
isPropagationStopped	Whether StopPropagation() was called for this event.
originalMousePosition	The original mouse position of the IMGUI event, before it is transformed

	to the current target local coordinates.
pooled	Whether the event is allocated from a pool of events.
propagationPhase	The current propagation phase.
target	The target visual element that received this event. Unlike currentTarget, this target does not change when the event is sent to other elements along the propagation path.
timestamp	The time when the event was created.
tricklesDown	Whether this event is sent down the event propagation path during the TrickleDown phase.

Public Methods

Dispose	Implementation of IDisposable.
GetEventTypeId	Retrieves the type id for this event instance.
PreventDefault	Whether the default actions are prevented from being executed for this event.
StopImmediatePropagation	Immediately stops the propagation of the event. The event is not sent to other elements along the propagation path. This method prevents other event handlers from executing on the current target.
StopPropagation	Stops propagating this event. The event is not sent to other elements along the propagation path. This method does not prevent other event handlers from executing on the current target.
Dispose	Implementation of IDispose.
GetEventTypeId	Retrieves the type id for this event instance.

Static Methods

RegisterEventType	Registers an event class to the event type system.
GetPooled	Gets an event from the event pool. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().
TypeId	Gets the type id for the event class.

Description

Sends an event when text from a TextField changes.

Properties

newData	The new text.
previousData	The text before the change occured.

Constructors

InputEvent	Constructor.

Protected Methods

Init	Resets the event members to their initial values.

Static Methods

GetPooled	Gets an event from the event pool and initializes it with the given values. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().

Inherited Members

Properties

bubbles	Whether this event type bubbles up in the event propagation path.
currentTarget	The current target of the event. The current target is the element in the propagation path for which event handlers are currently being executed.
dispatch	Whether the event is being dispatched to a visual element. An event cannot be redispatched while it being dispatched. If you need to recursively dispatch an event, it is recommended that you use a copy of the event.
flags	Flags for the event.
imguiEvent	The IMGUIEvent at the source of this event. The source can be null since not all events are generated by IMGUI.

isDefaultPrevented	Return true if the default actions should not be executed for this event.
isImmediatePropagationStopped	Whether StopImmediatePropagation() was called for this event.
isPropagationStopped	Whether StopPropagation() was called for this event.
originalMousePosition	The original mouse position of the IMGUI event, before it is transformed to the current target local coordinates.
pooled	Whether the event is allocated from a pool of events.
propagationPhase	The current propagation phase.
target	The target visual element that received this event. Unlike currentTarget, this target does not change when the event is sent to other elements along the propagation path.
timestamp	The time when the event was created.
tricklesDown	Whether this event is sent down the event propagation path during the TrickleDown phase.

Public Methods

Dispose	Implementation of IDisposable.
GetEventTypeId	Retrieves the type id for this event instance.
PreventDefault	Whether the default actions are prevented from being executed for this event.
StopImmediatePropagation	Immediately stops the propagation of the event. The event is not sent to other elements along the propagation path. This method prevents other event handlers from executing on the current target.
StopPropagation	Stops propagating this event. The event is not sent to other elements along the propagation path. This method does not prevent other event handlers from executing on the current target.
Dispose	Implementation of IDispose.
GetEventTypeId	Retrieves the type id for this event instance.

Static Methods

RegisterEventType	Registers an event class to the event type system.
GetPooled	Gets an event from the event pool. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().
TypeId	Gets the type id for the event class.

Implements interfaces:IKeyboardEvent

Description

Base class for keyboard events.

Properties

actionKey	Returns true if the platform specific action key is pressed. This key is Command on macOS and Control otherwise.
altKey	Returns true if the Alt key is pressed.
character	The character.
commandKey	Returns true if the Windows/Command key is pressed.
ctrlKey	Returns true if the Control key is pressed.
keyCode	The key code.
modifiers	Flags holding the pressed modifier keys (Alt, Control, Shift, Windows/Command).
shiftKey	Returns true if the Shift key is pressed.

Protected Methods

Init	Resets the event members to their initial values.

Static Methods

GetPooled	Gets a keyboard event from the event pool and initializes it with the given values. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().

Inherited Members

Properties

bubbles	Whether this event type bubbles up in the event propagation path.
currentTarget	The current target of the event. The current target is the element in the propagation path for which event handlers are currently being executed.
dispatch	Whether the event is being dispatched to a visual element. An event

	cannot be redispatched while it being dispatched. If you need to recursively dispatch an event, it is recommended that you use a copy of the event.
flags	Flags for the event.
imguiEvent	The IMGUIEvent at the source of this event. The source can be null since not all events are generated by IMGUI.
isDefaultPrevented	Return true if the default actions should not be executed for this event.
isImmediatePropagationStopped	Whether StopImmediatePropagation() was called for this event.
isPropagationStopped	Whether StopPropagation() was called for this event.
originalMousePosition	The original mouse position of the IMGUI event, before it is transformed to the current target local coordinates.
pooled	Whether the event is allocated from a pool of events.
propagationPhase	The current propagation phase.
target	The target visual element that received this event. Unlike currentTarget, this target does not change when the event is sent to other elements along the propagation path.
timestamp	The time when the event was created.
tricklesDown	Whether this event is sent down the event propagation path during the TrickleDown phase.

Public Methods

Dispose	Implementation of IDisposable.
GetEventTypeId	Retrieves the type id for this event instance.
PreventDefault	Whether the default actions are prevented from being executed for this event.
StopImmediatePropagation	Immediately stops the propagation of the event. The event is not sent to other elements along the propagation path. This method prevents other event handlers from executing on the current target.
StopPropagation	Stops propagating this event. The event is not sent to other elements along the propagation path. This method does not prevent other event handlers from executing on the current target.
Dispose	Implementation of IDispose.
GetEventTypeId	Retrieves the type id for this event instance.

Static Methods

RegisterEventType	Registers an event class to the event type system.
GetPooled	Gets an event from the event pool. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().
TypeId	Gets the type id for the event class.

Description

Event sent when a key is pressed on the keyboard. This event trickles down and bubbles up. This event is cancellable.

Constructors

KeyDownEvent	Constructor. Avoid newing events. Instead, use GetPooled() to get an event from a pool of reusable events.

Inherited Members

Properties

bubbles	Whether this event type bubbles up in the event propagation path.
currentTarget	The current target of the event. The current target is the element in the propagation path for which event handlers are currently being executed.
dispatch	Whether the event is being dispatched to a visual element. An event cannot be redispatched while it being dispatched. If you need to recursively dispatch an event, it is recommended that you use a copy of the event.
flags	Flags for the event.
imguiEvent	The IMGUIEvent at the source of this event. The source can be null since not all events are generated by IMGUI.
isDefaultPrevented	Return true if the default actions should not be executed for this event.
isImmediatePropagationStopped	Whether StopImmediatePropagation() was called for this event.
isPropagationStopped	Whether StopPropagation() was called for this event.
originalMousePosition	The original mouse position of the IMGUI event, before it is transformed to the current target local coordinates.
pooled	Whether the event is allocated from a pool of events.
propagationPhase	The current propagation phase.
target	The target visual element that received this event. Unlike currentTarget, this target does not change when the event is sent to other elements along the propagation path.
timestamp	The time when the event was created.

tricklesDown	Whether this event is sent down the event propagation path during the TrickleDown phase.
actionKey	Returns true if the platform specific action key is pressed. This key is Command on macOS and Control otherwise.
altKey	Returns true if the Alt key is pressed.
character	The character.
commandKey	Returns true if the Windows/Command key is pressed.
ctrlKey	Returns true if the Control key is pressed.
keyCode	The key code.
modifiers	Flags holding the pressed modifier keys (Alt, Control, Shift, Windows/Command).
shiftKey	Returns true if the Shift key is pressed.

Public Methods

Dispose	Implementation of IDisposable.
GetEventTypeId	Retrieves the type id for this event instance.
PreventDefault	Whether the default actions are prevented from being executed for this event.
StopImmediatePropagation	Immediately stops the propagation of the event. The event is not sent to other elements along the propagation path. This method prevents other event handlers from executing on the current target.
StopPropagation	Stops propagating this event. The event is not sent to other elements along the propagation path. This method does not prevent other event handlers from executing on the current target.
Dispose	Implementation of IDispose.
GetEventTypeId	Retrieves the type id for this event instance.

Protected Methods

Init	Resets all event members to their initial values.
Init	Resets all event members to their initial values.
Init	Resets the event members to their initial values.

Static Methods

RegisterEventType	Registers an event class to the event type system.
GetPooled	Gets an event from the event pool. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().
TypeId	Gets the type id for the event class.
GetPooled	Gets a keyboard event from the event pool and initializes it with the given values. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().

Description

Event sent when a key is released on the keyboard. This event trickles down and bubbles up. This event is cancellable.

Constructors

KeyUpEvent	Constructor. Avoid newing events. Instead, use GetPooled() to get an event from a pool of reusable events.

Inherited Members

Properties

bubbles	Whether this event type bubbles up in the event propagation path.
currentTarget	The current target of the event. The current target is the element in the propagation path for which event handlers are currently being executed.
dispatch	Whether the event is being dispatched to a visual element. An event cannot be redispatched while it being dispatched. If you need to recursively dispatch an event, it is recommended that you use a copy of the event.
flags	Flags for the event.
imguiEvent	The IMGUIEvent at the source of this event. The source can be null since not all events are generated by IMGUI.
isDefaultPrevented	Return true if the default actions should not be executed for this event.
isImmediatePropagationStopped	Whether StopImmediatePropagation() was called for this event.
isPropagationStopped	Whether StopPropagation() was called for this event.
originalMousePosition	The original mouse position of the IMGUI event, before it is transformed to the current target local coordinates.
pooled	Whether the event is allocated from a pool of events.
propagationPhase	The current propagation phase.
target	The target visual element that received this event. Unlike currentTarget, this target does not change when the event is sent to other elements along the propagation path.
timestamp	The time when the event was created.

tricklesDown	Whether this event is sent down the event propagation path during the TrickleDown phase.
actionKey	Returns true if the platform specific action key is pressed. This key is Command on macOS and Control otherwise.
altKey	Returns true if the Alt key is pressed.
character	The character.
commandKey	Returns true if the Windows/Command key is pressed.
ctrlKey	Returns true if the Control key is pressed.
keyCode	The key code.
modifiers	Flags holding the pressed modifier keys (Alt, Control, Shift, Windows/Command).
shiftKey	Returns true if the Shift key is pressed.

Public Methods

Dispose	Implementation of IDisposable.
GetEventTypeId	Retrieves the type id for this event instance.
PreventDefault	Whether the default actions are prevented from being executed for this event.
StopImmediatePropagation	Immediately stops the propagation of the event. The event is not sent to other elements along the propagation path. This method prevents other event handlers from executing on the current target.
StopPropagation	Stops propagating this event. The event is not sent to other elements along the propagation path. This method does not prevent other event handlers from executing on the current target.
Dispose	Implementation of IDispose.
GetEventTypeId	Retrieves the type id for this event instance.

Protected Methods

Init	Resets all event members to their initial values.
Init	Resets all event members to their initial values.
Init	Resets the event members to their initial values.

Static Methods

RegisterEventType	Registers an event class to the event type system.
GetPooled	Gets an event from the event pool. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().
TypeId	Gets the type id for the event class.
GetPooled	Gets a keyboard event from the event pool and initializes it with the given values. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().

Description

A vertically scrollable area that only creates visual elements for visible items while allowing the binding of many more items. As the user scrolls, visual elements are recycled and re-bound to new data items.

Properties

bindItem	Callback for binding a data item to the visual element.
itemHeight	ListView requires all visual elements to have the same height so that it can calculate a sensible scroller size. This property must be set for the list view to function.
itemsSource	The items data source. This property must be set for the list view to function.
makeItem	Callback for constructing the VisualElement that will serve as the template for each recycled and re-bound element in the list. This property must be set for the list view to function.
selectedIndex	Currently selected item index in the items source. If multiple items are selected, this will return the first selected item's index.
selectedItem	The currently selected item from the items source. If multiple items are selected, this will return the first selected item.
selectionType	Controls the selection state, whether: selections are disabled, there is only one selectable item, or if there are multiple selectable items.

Public Methods

Refresh	Clear, recreate all visible visual elements, and rebind all items. This should be called whenever the items source changes.
ScrollTo	Scroll to a specific visual element.
ScrollToItem	Scroll so that a specific item index from the items source is visible.

Events

onItemChosen	Callback for when an item is chosen (double-click). This is different from just a selection.
onSelectionChanged	Callback for a selection change.

Inherited Members

Static Properties

defaultFocusIndex	The default focus index for newly created elements.

Properties

canGrabFocus	Return true if the element can be focused.
focusController	Return the focus controller for this element.
focusIndex	An integer used to sort focusables in the focus ring. A negative value means that the element can not be focused.
childCount	Number of child elements in this object's contentContainer
clippingOptions	Should this element clip painting to its boundaries.
contentContainer	child elements are added to this element, usually this
dataWatch	Access to this element data watch interface.
enabledInHierarchy	Returns true if the VisualElement is enabled in its own hierarchy.
enabledSelf	Returns true if the VisualElement is enabled locally.
persistenceKey	Used for view data persistence (ie. tree expanded states, scroll position, zoom level).
schedule	Retrieves this VisualElement's IVisualElementScheduler
shadow	Access to this element physical hierarchy
style	Reference to the style object of this element.
this[int]	Access to this element physical hierarchy
tooltip	Text to display inside an information box after the user hovers the element for a small amount of time.
userData	This property can be used to associate application-specific user data with this VisualElement.

Public Methods

HandleEvent	Handle an event, most often by executing the callbacks associated with the event.
HasBubbleUpHandlers	Return true if event handlers for the event propagation BubbleUp phase have been attached on this object.
HasTrickleDownHandlers	Returns true if event handlers, for the event propagation TrickleDown phase, are attached to this object.

RegisterCallback	Adds an event handler to the instance. If the event handler has already been registered for the same phase (either TrickleDown or BubbleUp) then this method has no effect.
SendEvent	Sends an event to the event handler.
UnregisterCallback	Remove callback from the instance.
Blur	Tell the element to release the focus.
Focus	Attempt to give the focus to this element.
Add	Add an element to this element's contentContainer
AddStyleSheetPath	Adds this stylesheet file to this element list of applied styles
BringToFront	Brings this element to the end of its parent children list. The element will be visually in front of any overlapping sibling elements.
Children	Returns the elements from its contentContainer
Clear	Remove all child elements from this element's contentContainer
Contains	Returns true if the element is a direct child of this VisualElement
ElementAt	Retrieves the child element at position
EnableInClassList	Enables or disables the class with the given name.
FindAncestorUserData	Searchs up the hierachy of this VisualElement and retrieves stored userData, if any is found.
FindCommonAncestor	Finds the lowest commont ancestor between two VisualElements inside the VisualTree hierarchy
GetEnumerator	Allows to iterate into this elements children
GetFirstAncestorOfType	Walks up the hierarchy, starting from this element's parent, and returns the first VisualElement of this type
GetFirstOfType	Walks up the hierarchy, starting from this element, and returns the first VisualElement of this type
GetFullHierarchicalPersistenceKey	Combine this VisualElement's VisualElement.persistenceKey with those of its parents to create a more unique key for use with VisualElement.GetOrCreatePersistentData.
GetOrCreatePersistentData	Takes a reference to an existing persisted object and a key and returns the object either filled with the persisted state or as-is.
HasStyleSheetPath	Checks if this stylesheet file is in this element list of applied styles
IndexOf	Retrieves the child index of the specified VisualElement.

Insert	Insert an element into this element's contentContainer
MarkDirtyRepaint	Triggers a repaint of the VisualElement on the next frame.
OnPersistentDataReady	Called when the persistent data is accessible and/or when the data or persistence key have changed (VisualElement is properly parented).
OverwriteFromPersistedData	Overwrite object from the persistent data store.
PlaceBehind	Places this element right before the sibling element in their parent children list. If the element and the sibling position overlap, the element will be visually behind of its sibling.
PlaceInFront	Places this element right after the sibling element in their parent children list. If the element and the sibling position overlap, the element will be visually in front of its sibling.
Remove	Removes this child from the hierarchy
RemoveAt	Remove the child element located at this position from this element's contentContainer
RemoveFromHierarchy	Removes this element from its parent hierarchy
RemoveStyleSheetPath	Removes this stylesheet file from this element list of applied styles
SavePersistentData	Write persistence data to file.
SendEvent	Sends an event to the event handler.
SendToBack	Sends this element to the beginning of its parent children list. The element will be visually behind any overlapping sibling elements.
SetEnabled	Changes whether the current VisualElement is enabled or not. When disabled, a VisualElement does not receive most events.
Sort	Reorders child elements from this VisualElement contentContainer.
ToggleInClassList	Toggles between adding and removing the given class name from the class list.

Protected Methods

OnStyleResolved	Callback when the styles of an object have changed.

Description

Used by manipulators to match events against their requirements.

```
using UnityEditor;
using UnityEngine;
using UnityEngine.Experimental.UIElements;


public class ClickableTest
{
    public void CreateClickable()
    {
        var clickable = new Clickable(() => { Debug.Log("Clicked!"); });
        clickable.activators.Add(new ManipulatorActivationFilter { button = Mous
        clickable.activators.Add(new ManipulatorActivationFilter { button = Mous
    }
}
```

Properties

button	The button that activates the manipulation.
clickCount	Number of mouse clicks required to activate the manipulator.
modifiers	Any modifier keys (ie. ctrl, alt, ...) that are needed to activate the manipulation.

Public Methods

Matches	Returns true if the current mouse event satisfies the activation requirements.

Description

Class that manages capturing mouse events.

Static Methods

CaptureMouse	Assigns an event handler to capture mouse events.
HasMouseCapture	Checks if the event handler is capturing the mouse.
IsMouseCaptured	Checks if there is a handler capturing the mouse.
ReleaseMouse	Stops an event handler from capturing the mouse.

Description

Event sent after a handler starts capturing the mouse.

Constructors

MouseCaptureEvent	Constructor.

Inherited Members

Properties

bubbles	Whether this event type bubbles up in the event propagation path.
currentTarget	The current target of the event. The current target is the element in the propagation path for which event handlers are currently being executed.
dispatch	Whether the event is being dispatched to a visual element. An event cannot be redispatched while it being dispatched. If you need to recursively dispatch an event, it is recommended that you use a copy of the event.
flags	Flags for the event.
imguiEvent	The IMGUIEvent at the source of this event. The source can be null since not all events are generated by IMGUI.
isDefaultPrevented	Return true if the default actions should not be executed for this event.
isImmediatePropagationStopped	Whether StopImmediatePropagation() was called for this event.
isPropagationStopped	Whether StopPropagation() was called for this event.
originalMousePosition	The original mouse position of the IMGUI event, before it is transformed to the current target local coordinates.
pooled	Whether the event is allocated from a pool of events.
propagationPhase	The current propagation phase.
target	The target visual element that received this event. Unlike currentTarget, this target does not change when the event is sent to other elements along the propagation path.
timestamp	The time when the event was created.
tricklesDown	Whether this event is sent down the event propagation path during the

	TrickleDown phase.
relatedTarget	In the case of a MouseCaptureEvent, this property is the IEventHandler that loses the capture. In the case of a MouseCaptureOutEvent, this property is the IEventHandler that gains the capture.

Public Methods

Dispose	Implementation of IDisposable.
GetEventTypeId	Retrieves the type id for this event instance.
PreventDefault	Whether the default actions are prevented from being executed for this event.
StopImmediatePropagation	Immediately stops the propagation of the event. The event is not sent to other elements along the propagation path. This method prevents other event handlers from executing on the current target.
StopPropagation	Stops propagating this event. The event is not sent to other elements along the propagation path. This method does not prevent other event handlers from executing on the current target.
Dispose	Implementation of IDispose.
GetEventTypeId	Retrieves the type id for this event instance.

Protected Methods

Init	Resets all event members to their initial values.
Init	Resets all event members to their initial values.
Init	Resets the event members to their initial values.

Static Methods

RegisterEventType	Registers an event class to the event type system.
GetPooled	Gets an event from the event pool. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().
TypeId	Gets the type id for the event class.
GetPooled	Retrieves an event from the event pool. Use this method to retrieve a mouse event and initialize the event, instead of creating a new mouse event. Events obtained from this method should be released back to the pool using Dispose().

Implements interfaces:IMouseCaptureEvent

Description

Event sent when the handler capturing the mouse changes.

Properties

relatedTarget	In the case of a MouseCaptureEvent, this property is the IEventHandler that loses the capture. In the case of a MouseCaptureOutEvent, this property is the IEventHandler that gains the capture.

Protected Methods

Init	Resets the event members to their initial values.

Static Methods

GetPooled	Retrieves an event from the event pool. Use this method to retrieve a mouse event and initialize the event, instead of creating a new mouse event. Events obtained from this method should be released back to the pool using Dispose().

Inherited Members

Properties

bubbles	Whether this event type bubbles up in the event propagation path.
currentTarget	The current target of the event. The current target is the element in the propagation path for which event handlers are currently being executed.
dispatch	Whether the event is being dispatched to a visual element. An event cannot be redispatched while it being dispatched. If you need to recursively dispatch an event, it is recommended that you use a copy of the event.
flags	Flags for the event.
imguiEvent	The IMGUIEvent at the source of this event. The source can be null since not all events are generated by IMGUI.
isDefaultPrevented	Return true if the default actions should not be executed for this event.
isImmediatePropagationStopped	Whether StopImmediatePropagation() was called for this event.

isPropagationStopped	Whether StopPropagation() was called for this event.
originalMousePosition	The original mouse position of the IMGUI event, before it is transformed to the current target local coordinates.
pooled	Whether the event is allocated from a pool of events.
propagationPhase	The current propagation phase.
target	The target visual element that received this event. Unlike currentTarget, this target does not change when the event is sent to other elements along the propagation path.
timestamp	The time when the event was created.
tricklesDown	Whether this event is sent down the event propagation path during the TrickleDown phase.

Public Methods

Dispose	Implementation of IDisposable.
GetEventTypeId	Retrieves the type id for this event instance.
PreventDefault	Whether the default actions are prevented from being executed for this event.
StopImmediatePropagation	Immediately stops the propagation of the event. The event is not sent to other elements along the propagation path. This method prevents other event handlers from executing on the current target.
StopPropagation	Stops propagating this event. The event is not sent to other elements along the propagation path. This method does not prevent other event handlers from executing on the current target.
Dispose	Implementation of IDispose.
GetEventTypeId	Retrieves the type id for this event instance.

Static Methods

RegisterEventType	Registers an event class to the event type system.
GetPooled	Gets an event from the event pool. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().
TypeId	Gets the type id for the event class.

Description

Event sent before a handler stops capturing the mouse.

Constructors

MouseCaptureOutEvent	Constructor.

Inherited Members

Properties

bubbles	Whether this event type bubbles up in the event propagation path.
currentTarget	The current target of the event. The current target is the element in the propagation path for which event handlers are currently being executed.
dispatch	Whether the event is being dispatched to a visual element. An event cannot be redispatched while it being dispatched. If you need to recursively dispatch an event, it is recommended that you use a copy of the event.
flags	Flags for the event.
imguiEvent	The IMGUIEvent at the source of this event. The source can be null since not all events are generated by IMGUI.
isDefaultPrevented	Return true if the default actions should not be executed for this event.
isImmediatePropagationStopped	Whether StopImmediatePropagation() was called for this event.
isPropagationStopped	Whether StopPropagation() was called for this event.
originalMousePosition	The original mouse position of the IMGUI event, before it is transformed to the current target local coordinates.
pooled	Whether the event is allocated from a pool of events.
propagationPhase	The current propagation phase.
target	The target visual element that received this event. Unlike currentTarget, this target does not change when the event is sent to other elements along the propagation path.
timestamp	The time when the event was created.
tricklesDown	Whether this event is sent down the event propagation path during the

	TrickleDown phase.
relatedTarget	In the case of a MouseCaptureEvent, this property is the IEventHandler that loses the capture. In the case of a MouseCaptureOutEvent, this property is the IEventHandler that gains the capture.

Public Methods

Dispose	Implementation of IDisposable.
GetEventTypeId	Retrieves the type id for this event instance.
PreventDefault	Whether the default actions are prevented from being executed for this event.
StopImmediatePropagation	Immediately stops the propagation of the event. The event is not sent to other elements along the propagation path. This method prevents other event handlers from executing on the current target.
StopPropagation	Stops propagating this event. The event is not sent to other elements along the propagation path. This method does not prevent other event handlers from executing on the current target.
Dispose	Implementation of IDispose.
GetEventTypeId	Retrieves the type id for this event instance.

Protected Methods

Init	Resets all event members to their initial values.
Init	Resets all event members to their initial values.
Init	Resets the event members to their initial values.

Static Methods

RegisterEventType	Registers an event class to the event type system.
GetPooled	Gets an event from the event pool. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().
TypeId	Gets the type id for the event class.
GetPooled	Retrieves an event from the event pool. Use this method to retrieve a mouse event and initialize the event, instead of creating a new mouse event. Events obtained from this method should be released back to the pool using Dispose().

Description

Mouse down event.

Constructors

MouseDownEvent	Constructor. Avoid newing events. Instead, use GetPooled() to get an event from a pool of reusable events.

Inherited Members

Properties

bubbles	Whether this event type bubbles up in the event propagation path.
currentTarget	The current target of the event. The current target is the element in the propagation path for which event handlers are currently being executed.
dispatch	Whether the event is being dispatched to a visual element. An event cannot be redispatched while it being dispatched. If you need to recursively dispatch an event, it is recommended that you use a copy of the event.
flags	Flags for the event.
imguiEvent	The IMGUIEvent at the source of this event. The source can be null since not all events are generated by IMGUI.
isDefaultPrevented	Return true if the default actions should not be executed for this event.
isImmediatePropagationStopped	Whether StopImmediatePropagation() was called for this event.
isPropagationStopped	Whether StopPropagation() was called for this event.
originalMousePosition	The original mouse position of the IMGUI event, before it is transformed to the current target local coordinates.
pooled	Whether the event is allocated from a pool of events.
propagationPhase	The current propagation phase.
target	The target visual element that received this event. Unlike currentTarget, this target does not change when the event is sent to other elements along the propagation path.
timestamp	The time when the event was created.

tricklesDown	Whether this event is sent down the event propagation path during the TrickleDown phase.
actionKey	Returns true if the platform specific action key is pressed. This key is Command on macOS and Control otherwise.
altKey	Returns true if the Alt key is pressed.
button	Integer representing the pressed mouse button: 0 is left, 1 is right, 2 is center.
clickCount	Number of clicks.
commandKey	Returns true if the Windows/Command key is pressed.
ctrlKey	Returns true if the Control key is pressed.
currentTarget	The current target of the event. The current target is the element in the propagation path for which event handlers are currently being executed.
localMousePosition	The mouse position in the current target coordinate system.
modifiers	Flags holding the pressed modifier keys (Alt, Control, Shift, Windows/Command).
mouseDelta	The difference of the mouse position between the previous mouse event and the current mouse event.
mousePosition	The mouse position in the screen coordinate system.
shiftKey	Returns true if the Shift key is pressed.

Public Methods

Dispose	Implementation of IDisposable.
GetEventTypeId	Retrieves the type id for this event instance.
PreventDefault	Whether the default actions are prevented from being executed for this event.
StopImmediatePropagation	Immediately stops the propagation of the event. The event is not sent to other elements along the propagation path. This method prevents other event handlers from executing on the current target.
StopPropagation	Stops propagating this event. The event is not sent to other elements along the propagation path. This method does not prevent other event handlers from executing on the current target.
Dispose	Implementation of IDispose.
GetEventTypeId	Retrieves the type id for this event instance.

Protected Methods

Init	Resets all event members to their initial values.
Init	Resets all event members to their initial values.
Init	Resets the event members to their initial values.

Static Methods

RegisterEventType	Registers an event class to the event type system.
GetPooled	Gets an event from the event pool. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().
TypeId	Gets the type id for the event class.
GetPooled	Gets an event from the event pool and initializes it with the given values. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().

Description

Event sent when the mouse pointer enters an element or one of its descendent elements. The event is cancellable, it does not trickle down, and it does not bubble up.

Constructors

MouseEnterEvent	Constructor. Avoid newing events. Instead, use GetPooled() to get an event from a pool of reusable events.

Protected Methods

Init	Resets the event members to their initial values.

Inherited Members

Properties

bubbles	Whether this event type bubbles up in the event propagation path.
currentTarget	The current target of the event. The current target is the element in the propagation path for which event handlers are currently being executed.
dispatch	Whether the event is being dispatched to a visual element. An event cannot be redispatched while it being dispatched. If you need to recursively dispatch an event, it is recommended that you use a copy of the event.
flags	Flags for the event.
imguiEvent	The IMGUIEvent at the source of this event. The source can be null since not all events are generated by IMGUI.
isDefaultPrevented	Return true if the default actions should not be executed for this event.
isImmediatePropagationStopped	Whether StopImmediatePropagation() was called for this event.
isPropagationStopped	Whether StopPropagation() was called for this event.
originalMousePosition	The original mouse position of the IMGUI event, before it is transformed to the current target local coordinates.
pooled	Whether the event is allocated from a pool of events.
propagationPhase	The current propagation phase.

target	The target visual element that received this event. Unlike currentTarget, this target does not change when the event is sent to other elements along the propagation path.
timestamp	The time when the event was created.
tricklesDown	Whether this event is sent down the event propagation path during the TrickleDown phase.
actionKey	Returns true if the platform specific action key is pressed. This key is Command on macOS and Control otherwise.
altKey	Returns true if the Alt key is pressed.
button	Integer representing the pressed mouse button: 0 is left, 1 is right, 2 is center.
clickCount	Number of clicks.
commandKey	Returns true if the Windows/Command key is pressed.
ctrlKey	Returns true if the Control key is pressed.
currentTarget	The current target of the event. The current target is the element in the propagation path for which event handlers are currently being executed.
localMousePosition	The mouse position in the current target coordinate system.
modifiers	Flags holding the pressed modifier keys (Alt, Control, Shift, Windows/Command).
mouseDelta	The difference of the mouse position between the previous mouse event and the current mouse event.
mousePosition	The mouse position in the screen coordinate system.
shiftKey	Returns true if the Shift key is pressed.

Public Methods

Dispose	Implementation of IDisposable.
GetEventTypeId	Retrieves the type id for this event instance.
PreventDefault	Whether the default actions are prevented from being executed for this event.
StopImmediatePropagation	Immediately stops the propagation of the event. The event is not sent to other elements along the propagation path. This method prevents other event handlers from executing on the current target.
StopPropagation	Stops propagating this event. The event is not sent to other elements along the propagation path. This method does not prevent other event handlers from

	executing on the current target.
Dispose	Implementation of IDispose.
GetEventTypeId	Retrieves the type id for this event instance.

Static Methods

RegisterEventType	Registers an event class to the event type system.
GetPooled	Gets an event from the event pool. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().
TypeId	Gets the type id for the event class.
GetPooled	Gets an event from the event pool and initializes it with the given values. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().

Description

Event sent when the mouse pointer enters a window. The event is cancellable, it does not trickle down, and it does not bubble up.

Constructors

MouseEnterWindowEvent	Constructor. Avoid newing events. Instead, use GetPooled() to get an event from a pool of reusable events.

Protected Methods

Init	Resets the event members to their initial values.

Inherited Members

Properties

bubbles	Whether this event type bubbles up in the event propagation path.
currentTarget	The current target of the event. The current target is the element in the propagation path for which event handlers are currently being executed.
dispatch	Whether the event is being dispatched to a visual element. An event cannot be redispatched while it being dispatched. If you need to recursively dispatch an event, it is recommended that you use a copy of the event.
flags	Flags for the event.
imguiEvent	The IMGUIEvent at the source of this event. The source can be null since not all events are generated by IMGUI.
isDefaultPrevented	Return true if the default actions should not be executed for this event.
isImmediatePropagationStopped	Whether StopImmediatePropagation() was called for this event.
isPropagationStopped	Whether StopPropagation() was called for this event.
originalMousePosition	The original mouse position of the IMGUI event, before it is transformed to the current target local coordinates.
pooled	Whether the event is allocated from a pool of events.
propagationPhase	The current propagation phase.

target	The target visual element that received this event. Unlike currentTarget, this target does not change when the event is sent to other elements along the propagation path.
timestamp	The time when the event was created.
tricklesDown	Whether this event is sent down the event propagation path during the TrickleDown phase.
actionKey	Returns true if the platform specific action key is pressed. This key is Command on macOS and Control otherwise.
altKey	Returns true if the Alt key is pressed.
button	Integer representing the pressed mouse button: 0 is left, 1 is right, 2 is center.
clickCount	Number of clicks.
commandKey	Returns true if the Windows/Command key is pressed.
ctrlKey	Returns true if the Control key is pressed.
currentTarget	The current target of the event. The current target is the element in the propagation path for which event handlers are currently being executed.
localMousePosition	The mouse position in the current target coordinate system.
modifiers	Flags holding the pressed modifier keys (Alt, Control, Shift, Windows/Command).
mouseDelta	The difference of the mouse position between the previous mouse event and the current mouse event.
mousePosition	The mouse position in the screen coordinate system.
shiftKey	Returns true if the Shift key is pressed.

Public Methods

Dispose	Implementation of IDisposable.
GetEventTypeId	Retrieves the type id for this event instance.
PreventDefault	Whether the default actions are prevented from being executed for this event.
StopImmediatePropagation	Immediately stops the propagation of the event. The event is not sent to other elements along the propagation path. This method prevents other event handlers from executing on the current target.
StopPropagation	Stops propagating this event. The event is not sent to other elements along the propagation path. This method does not prevent other event handlers from

	executing on the current target.
Dispose	Implementation of IDispose.
GetEventTypeId	Retrieves the type id for this event instance.

Static Methods

RegisterEventType	Registers an event class to the event type system.
GetPooled	Gets an event from the event pool. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().
TypeId	Gets the type id for the event class.
GetPooled	Gets an event from the event pool and initializes it with the given values. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().

Implements interfaces:IMouseEvent

Description

The base class for mouse events.

Properties

actionKey	Returns true if the platform specific action key is pressed. This key is Command on macOS and Control otherwise.
altKey	Returns true if the Alt key is pressed.
button	Integer representing the pressed mouse button: 0 is left, 1 is right, 2 is center.
clickCount	Number of clicks.
commandKey	Returns true if the Windows/Command key is pressed.
ctrlKey	Returns true if the Control key is pressed.
currentTarget	The current target of the event. The current target is the element in the propagation path for which event handlers are currently being executed.
localMousePosition	The mouse position in the current target coordinate system.
modifiers	Flags holding the pressed modifier keys (Alt, Control, Shift, Windows/Command).
mouseDelta	The difference of the mouse position between the previous mouse event and the current mouse event.
mousePosition	The mouse position in the screen coordinate system.
shiftKey	Returns true if the Shift key is pressed.

Protected Methods

Init	Resets the event members to their initial values.

Static Methods

GetPooled	Gets an event from the event pool and initializes it with the given values. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().

Inherited Members

Properties

bubbles	Whether this event type bubbles up in the event propagation path.
currentTarget	The current target of the event. The current target is the element in the propagation path for which event handlers are currently being executed.
dispatch	Whether the event is being dispatched to a visual element. An event cannot be redispatched while it being dispatched. If you need to recursively dispatch an event, it is recommended that you use a copy of the event.
flags	Flags for the event.
imguiEvent	The IMGUIEvent at the source of this event. The source can be null since not all events are generated by IMGUI.
isDefaultPrevented	Return true if the default actions should not be executed for this event.
isImmediatePropagationStopped	Whether StopImmediatePropagation() was called for this event.
isPropagationStopped	Whether StopPropagation() was called for this event.
originalMousePosition	The original mouse position of the IMGUI event, before it is transformed to the current target local coordinates.
pooled	Whether the event is allocated from a pool of events.
propagationPhase	The current propagation phase.
target	The target visual element that received this event. Unlike currentTarget, this target does not change when the event is sent to other elements along the propagation path.
timestamp	The time when the event was created.
tricklesDown	Whether this event is sent down the event propagation path during the TrickleDown phase.

Public Methods

Dispose	Implementation of IDisposable.
GetEventTypeId	Retrieves the type id for this event instance.
PreventDefault	Whether the default actions are prevented from being executed for this event.
StopImmediatePropagation	Immediately stops the propagation of the event. The event is not sent to other elements along the propagation path. This method prevents other event

	handlers from executing on the current target.
StopPropagation	Stops propagating this event. The event is not sent to other elements along the propagation path. This method does not prevent other event handlers from executing on the current target.
Dispose	Implementation of IDispose.
GetEventTypeId	Retrieves the type id for this event instance.

Static Methods

RegisterEventType	Registers an event class to the event type system.
GetPooled	Gets an event from the event pool. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().
TypeId	Gets the type id for the event class.

Description

Event sent when the mouse pointer exits an element and all its descendent elements. The event is cancellable, it does not trickle down, and it does not bubble up.

Constructors

MouseLeaveEvent	Constructor. Avoid newing events. Instead, use GetPooled() to get an event from a pool of reusable events.

Protected Methods

Init	Resets the event members to their initial values.

Inherited Members

Properties

bubbles	Whether this event type bubbles up in the event propagation path.
currentTarget	The current target of the event. The current target is the element in the propagation path for which event handlers are currently being executed.
dispatch	Whether the event is being dispatched to a visual element. An event cannot be redispatched while it being dispatched. If you need to recursively dispatch an event, it is recommended that you use a copy of the event.
flags	Flags for the event.
imguiEvent	The IMGUIEvent at the source of this event. The source can be null since not all events are generated by IMGUI.
isDefaultPrevented	Return true if the default actions should not be executed for this event.
isImmediatePropagationStopped	Whether StopImmediatePropagation() was called for this event.
isPropagationStopped	Whether StopPropagation() was called for this event.
originalMousePosition	The original mouse position of the IMGUI event, before it is transformed to the current target local coordinates.
pooled	Whether the event is allocated from a pool of events.
propagationPhase	The current propagation phase.

target	The target visual element that received this event. Unlike currentTarget, this target does not change when the event is sent to other elements along the propagation path.
timestamp	The time when the event was created.
tricklesDown	Whether this event is sent down the event propagation path during the TrickleDown phase.
actionKey	Returns true if the platform specific action key is pressed. This key is Command on macOS and Control otherwise.
altKey	Returns true if the Alt key is pressed.
button	Integer representing the pressed mouse button: 0 is left, 1 is right, 2 is center.
clickCount	Number of clicks.
commandKey	Returns true if the Windows/Command key is pressed.
ctrlKey	Returns true if the Control key is pressed.
currentTarget	The current target of the event. The current target is the element in the propagation path for which event handlers are currently being executed.
localMousePosition	The mouse position in the current target coordinate system.
modifiers	Flags holding the pressed modifier keys (Alt, Control, Shift, Windows/Command).
mouseDelta	The difference of the mouse position between the previous mouse event and the current mouse event.
mousePosition	The mouse position in the screen coordinate system.
shiftKey	Returns true if the Shift key is pressed.

Public Methods

Dispose	Implementation of IDisposable.
GetEventTypeId	Retrieves the type id for this event instance.
PreventDefault	Whether the default actions are prevented from being executed for this event.
StopImmediatePropagation	Immediately stops the propagation of the event. The event is not sent to other elements along the propagation path. This method prevents other event handlers from executing on the current target.
StopPropagation	Stops propagating this event. The event is not sent to other elements along the propagation path. This method does not prevent other event handlers from

	executing on the current target.
Dispose	Implementation of IDispose.
GetEventTypeId	Retrieves the type id for this event instance.

Static Methods

RegisterEventType	Registers an event class to the event type system.
GetPooled	Gets an event from the event pool. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().
TypeId	Gets the type id for the event class.
GetPooled	Gets an event from the event pool and initializes it with the given values. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().

Description

Event sent when the mouse pointer exits a window. The event is cancellable, it does not trickle down, and it does not bubble up.

Constructors

MouseLeaveWindowEvent	Constructor. Avoid newing events. Instead, use GetPooled() to get an event from a pool of reusable events.

Protected Methods

Init	Resets the event members to their initial values.

Inherited Members

Properties

bubbles	Whether this event type bubbles up in the event propagation path.
currentTarget	The current target of the event. The current target is the element in the propagation path for which event handlers are currently being executed.
dispatch	Whether the event is being dispatched to a visual element. An event cannot be redispatched while it being dispatched. If you need to recursively dispatch an event, it is recommended that you use a copy of the event.
flags	Flags for the event.
imguiEvent	The IMGUIEvent at the source of this event. The source can be null since not all events are generated by IMGUI.
isDefaultPrevented	Return true if the default actions should not be executed for this event.
isImmediatePropagationStopped	Whether StopImmediatePropagation() was called for this event.
isPropagationStopped	Whether StopPropagation() was called for this event.
originalMousePosition	The original mouse position of the IMGUI event, before it is transformed to the current target local coordinates.
pooled	Whether the event is allocated from a pool of events.
propagationPhase	The current propagation phase.

target	The target visual element that received this event. Unlike currentTarget, this target does not change when the event is sent to other elements along the propagation path.
timestamp	The time when the event was created.
tricklesDown	Whether this event is sent down the event propagation path during the TrickleDown phase.
actionKey	Returns true if the platform specific action key is pressed. This key is Command on macOS and Control otherwise.
altKey	Returns true if the Alt key is pressed.
button	Integer representing the pressed mouse button: 0 is left, 1 is right, 2 is center.
clickCount	Number of clicks.
commandKey	Returns true if the Windows/Command key is pressed.
ctrlKey	Returns true if the Control key is pressed.
currentTarget	The current target of the event. The current target is the element in the propagation path for which event handlers are currently being executed.
localMousePosition	The mouse position in the current target coordinate system.
modifiers	Flags holding the pressed modifier keys (Alt, Control, Shift, Windows/Command).
mouseDelta	The difference of the mouse position between the previous mouse event and the current mouse event.
mousePosition	The mouse position in the screen coordinate system.
shiftKey	Returns true if the Shift key is pressed.

Public Methods

Dispose	Implementation of IDisposable.
GetEventTypeId	Retrieves the type id for this event instance.
PreventDefault	Whether the default actions are prevented from being executed for this event.
StopImmediatePropagation	Immediately stops the propagation of the event. The event is not sent to other elements along the propagation path. This method prevents other event handlers from executing on the current target.
StopPropagation	Stops propagating this event. The event is not sent to other elements along the propagation path. This method does not prevent other event handlers from

	executing on the current target.
Dispose	Implementation of IDispose.
GetEventTypeId	Retrieves the type id for this event instance.

Static Methods

RegisterEventType	Registers an event class to the event type system.
GetPooled	Gets an event from the event pool. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().
TypeId	Gets the type id for the event class.
GetPooled	Gets an event from the event pool and initializes it with the given values. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().

Description

Mouse move event.

Constructors

MouseMoveEvent	Constructor. Avoid newing events. Instead, use GetPooled() to get an event from a pool of reusable events.

Inherited Members

Properties

bubbles	Whether this event type bubbles up in the event propagation path.
currentTarget	The current target of the event. The current target is the element in the propagation path for which event handlers are currently being executed.
dispatch	Whether the event is being dispatched to a visual element. An event cannot be redispatched while it being dispatched. If you need to recursively dispatch an event, it is recommended that you use a copy of the event.
flags	Flags for the event.
imguiEvent	The IMGUIEvent at the source of this event. The source can be null since not all events are generated by IMGUI.
isDefaultPrevented	Return true if the default actions should not be executed for this event.
isImmediatePropagationStopped	Whether StopImmediatePropagation() was called for this event.
isPropagationStopped	Whether StopPropagation() was called for this event.
originalMousePosition	The original mouse position of the IMGUI event, before it is transformed to the current target local coordinates.
pooled	Whether the event is allocated from a pool of events.
propagationPhase	The current propagation phase.
target	The target visual element that received this event. Unlike currentTarget, this target does not change when the event is sent to other elements along the propagation path.
timestamp	The time when the event was created.

tricklesDown	Whether this event is sent down the event propagation path during the TrickleDown phase.
actionKey	Returns true if the platform specific action key is pressed. This key is Command on macOS and Control otherwise.
altKey	Returns true if the Alt key is pressed.
button	Integer representing the pressed mouse button: 0 is left, 1 is right, 2 is center.
clickCount	Number of clicks.
commandKey	Returns true if the Windows/Command key is pressed.
ctrlKey	Returns true if the Control key is pressed.
currentTarget	The current target of the event. The current target is the element in the propagation path for which event handlers are currently being executed.
localMousePosition	The mouse position in the current target coordinate system.
modifiers	Flags holding the pressed modifier keys (Alt, Control, Shift, Windows/Command).
mouseDelta	The difference of the mouse position between the previous mouse event and the current mouse event.
mousePosition	The mouse position in the screen coordinate system.
shiftKey	Returns true if the Shift key is pressed.

Public Methods

Dispose	Implementation of IDisposable.
GetEventTypeId	Retrieves the type id for this event instance.
PreventDefault	Whether the default actions are prevented from being executed for this event.
StopImmediatePropagation	Immediately stops the propagation of the event. The event is not sent to other elements along the propagation path. This method prevents other event handlers from executing on the current target.
StopPropagation	Stops propagating this event. The event is not sent to other elements along the propagation path. This method does not prevent other event handlers from executing on the current target.
Dispose	Implementation of IDispose.
GetEventTypeId	Retrieves the type id for this event instance.

Protected Methods

Init	Resets all event members to their initial values.
Init	Resets all event members to their initial values.
Init	Resets the event members to their initial values.

Static Methods

RegisterEventType	Registers an event class to the event type system.
GetPooled	Gets an event from the event pool. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().
TypeId	Gets the type id for the event class.
GetPooled	Gets an event from the event pool and initializes it with the given values. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().

Description

Event sent when the mouse pointer exits an element. The event trickles down, it bubbles up, and it is cancellable.

Constructors

MouseOutEvent	Constructor. Avoid newing events. Instead, use GetPooled() to get an event from a pool of reusable events.

Inherited Members

Properties

bubbles	Whether this event type bubbles up in the event propagation path.
currentTarget	The current target of the event. The current target is the element in the propagation path for which event handlers are currently being executed.
dispatch	Whether the event is being dispatched to a visual element. An event cannot be redispatched while it being dispatched. If you need to recursively dispatch an event, it is recommended that you use a copy of the event.
flags	Flags for the event.
imguiEvent	The IMGUIEvent at the source of this event. The source can be null since not all events are generated by IMGUI.
isDefaultPrevented	Return true if the default actions should not be executed for this event.
isImmediatePropagationStopped	Whether StopImmediatePropagation() was called for this event.
isPropagationStopped	Whether StopPropagation() was called for this event.
originalMousePosition	The original mouse position of the IMGUI event, before it is transformed to the current target local coordinates.
pooled	Whether the event is allocated from a pool of events.
propagationPhase	The current propagation phase.
target	The target visual element that received this event. Unlike currentTarget, this target does not change when the event is sent to other elements along the propagation path.
timestamp	The time when the event was created.

tricklesDown	Whether this event is sent down the event propagation path during the TrickleDown phase.
actionKey	Returns true if the platform specific action key is pressed. This key is Command on macOS and Control otherwise.
altKey	Returns true if the Alt key is pressed.
button	Integer representing the pressed mouse button: 0 is left, 1 is right, 2 is center.
clickCount	Number of clicks.
commandKey	Returns true if the Windows/Command key is pressed.
ctrlKey	Returns true if the Control key is pressed.
currentTarget	The current target of the event. The current target is the element in the propagation path for which event handlers are currently being executed.
localMousePosition	The mouse position in the current target coordinate system.
modifiers	Flags holding the pressed modifier keys (Alt, Control, Shift, Windows/Command).
mouseDelta	The difference of the mouse position between the previous mouse event and the current mouse event.
mousePosition	The mouse position in the screen coordinate system.
shiftKey	Returns true if the Shift key is pressed.

Public Methods

Dispose	Implementation of IDisposable.
GetEventTypeId	Retrieves the type id for this event instance.
PreventDefault	Whether the default actions are prevented from being executed for this event.
StopImmediatePropagation	Immediately stops the propagation of the event. The event is not sent to other elements along the propagation path. This method prevents other event handlers from executing on the current target.
StopPropagation	Stops propagating this event. The event is not sent to other elements along the propagation path. This method does not prevent other event handlers from executing on the current target.
Dispose	Implementation of IDispose.
GetEventTypeId	Retrieves the type id for this event instance.

Protected Methods

Init	Resets all event members to their initial values.
Init	Resets all event members to their initial values.
Init	Resets the event members to their initial values.

Static Methods

RegisterEventType	Registers an event class to the event type system.
GetPooled	Gets an event from the event pool. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().
TypeId	Gets the type id for the event class.
GetPooled	Gets an event from the event pool and initializes it with the given values. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().

Description

Event sent when the mouse pointer enters an element. The event trickles down, it bubbles up, and it is cancellable.

Constructors

MouseOverEvent	Constructor. Avoid newing events. Instead, use GetPooled() to get an event from a pool of reusable events.

Inherited Members

Properties

bubbles	Whether this event type bubbles up in the event propagation path.
currentTarget	The current target of the event. The current target is the element in the propagation path for which event handlers are currently being executed.
dispatch	Whether the event is being dispatched to a visual element. An event cannot be redispatched while it being dispatched. If you need to recursively dispatch an event, it is recommended that you use a copy of the event.
flags	Flags for the event.
imguiEvent	The IMGUIEvent at the source of this event. The source can be null since not all events are generated by IMGUI.
isDefaultPrevented	Return true if the default actions should not be executed for this event.
isImmediatePropagationStopped	Whether StopImmediatePropagation() was called for this event.
isPropagationStopped	Whether StopPropagation() was called for this event.
originalMousePosition	The original mouse position of the IMGUI event, before it is transformed to the current target local coordinates.
pooled	Whether the event is allocated from a pool of events.
propagationPhase	The current propagation phase.
target	The target visual element that received this event. Unlike currentTarget, this target does not change when the event is sent to other elements along the propagation path.
timestamp	The time when the event was created.

tricklesDown	Whether this event is sent down the event propagation path during the TrickleDown phase.
actionKey	Returns true if the platform specific action key is pressed. This key is Command on macOS and Control otherwise.
altKey	Returns true if the Alt key is pressed.
button	Integer representing the pressed mouse button: 0 is left, 1 is right, 2 is center.
clickCount	Number of clicks.
commandKey	Returns true if the Windows/Command key is pressed.
ctrlKey	Returns true if the Control key is pressed.
currentTarget	The current target of the event. The current target is the element in the propagation path for which event handlers are currently being executed.
localMousePosition	The mouse position in the current target coordinate system.
modifiers	Flags holding the pressed modifier keys (Alt, Control, Shift, Windows/Command).
mouseDelta	The difference of the mouse position between the previous mouse event and the current mouse event.
mousePosition	The mouse position in the screen coordinate system.
shiftKey	Returns true if the Shift key is pressed.

Public Methods

Dispose	Implementation of IDisposable.
GetEventTypeId	Retrieves the type id for this event instance.
PreventDefault	Whether the default actions are prevented from being executed for this event.
StopImmediatePropagation	Immediately stops the propagation of the event. The event is not sent to other elements along the propagation path. This method prevents other event handlers from executing on the current target.
StopPropagation	Stops propagating this event. The event is not sent to other elements along the propagation path. This method does not prevent other event handlers from executing on the current target.
Dispose	Implementation of IDispose.
GetEventTypeId	Retrieves the type id for this event instance.

Protected Methods

Init	Resets all event members to their initial values.
Init	Resets all event members to their initial values.
Init	Resets the event members to their initial values.

Static Methods

RegisterEventType	Registers an event class to the event type system.
GetPooled	Gets an event from the event pool. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().
TypeId	Gets the type id for the event class.
GetPooled	Gets an event from the event pool and initializes it with the given values. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().

Description

Mouse up event.

Constructors

MouseUpEvent	Constructor. Avoid newing events. Instead, use GetPooled() to get an event from a pool of reusable events.

Inherited Members

Properties

bubbles	Whether this event type bubbles up in the event propagation path.
currentTarget	The current target of the event. The current target is the element in the propagation path for which event handlers are currently being executed.
dispatch	Whether the event is being dispatched to a visual element. An event cannot be redispatched while it being dispatched. If you need to recursively dispatch an event, it is recommended that you use a copy of the event.
flags	Flags for the event.
imguiEvent	The IMGUIEvent at the source of this event. The source can be null since not all events are generated by IMGUI.
isDefaultPrevented	Return true if the default actions should not be executed for this event.
isImmediatePropagationStopped	Whether StopImmediatePropagation() was called for this event.
isPropagationStopped	Whether StopPropagation() was called for this event.
originalMousePosition	The original mouse position of the IMGUI event, before it is transformed to the current target local coordinates.
pooled	Whether the event is allocated from a pool of events.
propagationPhase	The current propagation phase.
target	The target visual element that received this event. Unlike currentTarget, this target does not change when the event is sent to other elements along the propagation path.
timestamp	The time when the event was created.

tricklesDown	Whether this event is sent down the event propagation path during the TrickleDown phase.
actionKey	Returns true if the platform specific action key is pressed. This key is Command on macOS and Control otherwise.
altKey	Returns true if the Alt key is pressed.
button	Integer representing the pressed mouse button: 0 is left, 1 is right, 2 is center.
clickCount	Number of clicks.
commandKey	Returns true if the Windows/Command key is pressed.
ctrlKey	Returns true if the Control key is pressed.
currentTarget	The current target of the event. The current target is the element in the propagation path for which event handlers are currently being executed.
localMousePosition	The mouse position in the current target coordinate system.
modifiers	Flags holding the pressed modifier keys (Alt, Control, Shift, Windows/Command).
mouseDelta	The difference of the mouse position between the previous mouse event and the current mouse event.
mousePosition	The mouse position in the screen coordinate system.
shiftKey	Returns true if the Shift key is pressed.

Public Methods

Dispose	Implementation of IDisposable.
GetEventTypeId	Retrieves the type id for this event instance.
PreventDefault	Whether the default actions are prevented from being executed for this event.
StopImmediatePropagation	Immediately stops the propagation of the event. The event is not sent to other elements along the propagation path. This method prevents other event handlers from executing on the current target.
StopPropagation	Stops propagating this event. The event is not sent to other elements along the propagation path. This method does not prevent other event handlers from executing on the current target.
Dispose	Implementation of IDispose.
GetEventTypeId	Retrieves the type id for this event instance.

Protected Methods

Init	Resets all event members to their initial values.
Init	Resets all event members to their initial values.
Init	Resets the event members to their initial values.

Static Methods

RegisterEventType	Registers an event class to the event type system.
GetPooled	Gets an event from the event pool. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().
TypeId	Gets the type id for the event class.
GetPooled	Gets an event from the event pool and initializes it with the given values. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().

Description

Styled visual element that matches the EditorGUILayout.Popup IMGUI element.

Inherited Members

Static Properties

defaultFocusIndex	The default focus index for newly created elements.

Properties

canGrabFocus	Return true if the element can be focused.
focusController	Return the focus controller for this element.
focusIndex	An integer used to sort focusables in the focus ring. A negative value means that the element can not be focused.
text	The text associated with the element.
childCount	Number of child elements in this object's contentContainer
clippingOptions	Should this element clip painting to its boundaries.
contentContainer	child elements are added to this element, usually this
dataWatch	Access to this element data watch interface.
enabledInHierarchy	Returns true if the VisualElement is enabled in its own hierarchy.
enabledSelf	Returns true if the VisualElement is enabled locally.
persistenceKey	Used for view data persistence (ie. tree expanded states, scroll position, zoom level).
schedule	Retrieves this VisualElement's IVisualElementScheduler
shadow	Access to this element physical hierarchy
style	Reference to the style object of this element.
this[int]	Access to this element physical hierarchy
tooltip	Text to display inside an information box after the user hovers the element for a small amount of time.
userData	This property can be used to associate application-specific user data with this VisualElement.

Public Methods

HandleEvent	Handle an event, most often by executing the callbacks associated with the event.
HasBubbleUpHandlers	Return true if event handlers for the event propagation BubbleUp phase have been attached on this object.
HasTrickleDownHandlers	Returns true if event handlers, for the event propagation TrickleDown phase, are attached to this object.
RegisterCallback	Adds an event handler to the instance. If the event handler has already been registered for the same phase (either TrickleDown or BubbleUp) then this method has no effect.
SendEvent	Sends an event to the event handler.
UnregisterCallback	Remove callback from the instance.
Blur	Tell the element to release the focus.
Focus	Attempt to give the focus to this element.
MeasureTextSize	Computes the size needed to display a text string based on element style values such as font, font-size, word-wrap, and so on.
Add	Add an element to this element's contentContainer
AddStyleSheetPath	Adds this stylesheet file to this element list of applied styles
BringToFront	Brings this element to the end of its parent children list. The element will be visually in front of any overlapping sibling elements.
Children	Returns the elements from its contentContainer
Clear	Remove all child elements from this element's contentContainer
Contains	Returns true if the element is a direct child of this VisualElement
ElementAt	Retrieves the child element at position
EnableInClassList	Enables or disables the class with the given name.
FindAncestorUserData	Searchs up the hierachy of this VisualElement and retrieves stored userData, if any is found.
FindCommonAncestor	Finds the lowest commont ancestor between two VisualElements inside the VisualTree hierarchy
GetEnumerator	Allows to iterate into this elements children
GetFirstAncestorOfType	Walks up the hierarchy, starting from this element's parent, and returns the first VisualElement of this type

GetFirstOfType	Walks up the hierarchy, starting from this element, and returns the first VisualElement of this type
GetFullHierarchicalPersistenceKey	Combine this VisualElement's VisualElement.persistenceKey with those of its parents to create a more unique key for use with VisualElement.GetOrCreatePersistentData.
GetOrCreatePersistentData	Takes a reference to an existing persisted object and a key and returns the object either filled with the persisted state or as-is.
HasStyleSheetPath	Checks if this stylesheet file is in this element list of applied styles
IndexOf	Retrieves the child index of the specified VisualElement.
Insert	Insert an element into this element's contentContainer
MarkDirtyRepaint	Triggers a repaint of the VisualElement on the next frame.
OnPersistentDataReady	Called when the persistent data is accessible and/or when the data or persistence key have changed (VisualElement is properly parented).
OverwriteFromPersistedData	Overwrite object from the persistent data store.
PlaceBehind	Places this element right before the sibling element in their parent children list. If the element and the sibling position overlap, the element will be visually behind of its sibling.
PlaceInFront	Places this element right after the sibling element in their parent children list. If the element and the sibling position overlap, the element will be visually in front of its sibling.
Remove	Removes this child from the hierarchy
RemoveAt	Remove the child element located at this position from this element's contentContainer
RemoveFromHierarchy	Removes this element from its parent hierarchy
RemoveStyleSheetPath	Removes this stylesheet file from this element list of applied styles
SavePersistentData	Write persistence data to file.
SendEvent	Sends an event to the event handler.
SendToBack	Sends this element to the beginning of its parent children list. The element will be visually behind any overlapping sibling elements.
SetEnabled	Changes whether the current VisualElement is enabled or not. When disabled, a VisualElement does not receive most events.
Sort	Reorders child elements from this VisualElement contentContainer.
ToggleInClassList	Toggles between adding and removing the given class name from the

class list.

Protected Methods

OnStyleResolved	Callback when the styles of an object have changed.

Description

A button that executes an action repeatedly while it is pressed.

Constructors

RepeatButton	Constructor.

Public Methods

SetAction	Set the action that should be executed when the button is pressed.

Inherited Members

Static Properties

defaultFocusIndex	The default focus index for newly created elements.

Properties

canGrabFocus	Return true if the element can be focused.
focusController	Return the focus controller for this element.
focusIndex	An integer used to sort focusables in the focus ring. A negative value means that the element can not be focused.
text	The text associated with the element.
childCount	Number of child elements in this object's contentContainer
clippingOptions	Should this element clip painting to its boundaries.
contentContainer	child elements are added to this element, usually this
dataWatch	Access to this element data watch interface.
enabledInHierarchy	Returns true if the VisualElement is enabled in its own hierarchy.
enabledSelf	Returns true if the VisualElement is enabled locally.
persistenceKey	Used for view data persistence (ie. tree expanded states, scroll position, zoom level).
schedule	Retrieves this VisualElement's IVisualElementScheduler

shadow	Access to this element physical hierarchy
style	Reference to the style object of this element.
this[int]	Access to this element physical hierarchy
tooltip	Text to display inside an information box after the user hovers the element for a small amount of time.
userData	This property can be used to associate application-specific user data with this VisualElement.

Public Methods

HandleEvent	Handle an event, most often by executing the callbacks associated with the event.
HasBubbleUpHandlers	Return true if event handlers for the event propagation BubbleUp phase have been attached on this object.
HasTrickleDownHandlers	Returns true if event handlers, for the event propagation TrickleDown phase, are attached to this object.
RegisterCallback	Adds an event handler to the instance. If the event handler has already been registered for the same phase (either TrickleDown or BubbleUp) then this method has no effect.
SendEvent	Sends an event to the event handler.
UnregisterCallback	Remove callback from the instance.
Blur	Tell the element to release the focus.
Focus	Attempt to give the focus to this element.
MeasureTextSize	Computes the size needed to display a text string based on element style values such as font, font-size, word-wrap, and so on.
Add	Add an element to this element's contentContainer
AddStyleSheetPath	Adds this stylesheet file to this element list of applied styles
BringToFront	Brings this element to the end of its parent children list. The element will be visually in front of any overlapping sibling elements.
Children	Returns the elements from its contentContainer
Clear	Remove all child elements from this element's contentContainer
Contains	Returns true if the element is a direct child of this VisualElement
ElementAt	Retrieves the child element at position

EnableInClassList	Enables or disables the class with the given name.
FindAncestorUserData	Searchs up the hierachy of this VisualElement and retrieves stored userData, if any is found.
FindCommonAncestor	Finds the lowest commont ancestor between two VisualElements inside the VisualTree hierarchy
GetEnumerator	Allows to iterate into this elements children
GetFirstAncestorOfType	Walks up the hierarchy, starting from this element's parent, and returns the first VisualElement of this type
GetFirstOfType	Walks up the hierarchy, starting from this element, and returns the first VisualElement of this type
GetFullHierarchicalPersistenceKey	Combine this VisualElement's VisualElement.persistenceKey with those of its parents to create a more unique key for use with VisualElement.GetOrCreatePersistentData.
GetOrCreatePersistentData	Takes a reference to an existing persisted object and a key and returns the object either filled with the persisted state or as-is.
HasStyleSheetPath	Checks if this stylesheet file is in this element list of applied styles
IndexOf	Retrieves the child index of the specified VisualElement.
Insert	Insert an element into this element's contentContainer
MarkDirtyRepaint	Triggers a repaint of the VisualElement on the next frame.
OnPersistentDataReady	Called when the persistent data is accessible and/or when the data or persistence key have changed (VisualElement is properly parented).
OverwriteFromPersistedData	Overwrite object from the persistent data store.
PlaceBehind	Places this element right before the sibling element in their parent children list. If the element and the sibling position overlap, the element will be visually behind of its sibling.
PlaceInFront	Places this element right after the sibling element in their parent children list. If the element and the sibling position overlap, the element will be visually in front of its sibling.
Remove	Removes this child from the hierarchy
RemoveAt	Remove the child element located at this position from this element's contentContainer
RemoveFromHierarchy	Removes this element from its parent hierarchy
RemoveStyleSheetPath	Removes this stylesheet file from this element list of applied styles

SavePersistentData	Write persistence data to file.
SendEvent	Sends an event to the event handler.
SendToBack	Sends this element to the beginning of its parent children list. The element will be visually behind any overlapping sibling elements.
SetEnabled	Changes whether the current VisualElement is enabled or not. When disabled, a VisualElement does not receive most events.
Sort	Reorders child elements from this VisualElement contentContainer.
ToggleInClassList	Toggles between adding and removing the given class name from the class list.

Protected Methods

OnStyleResolved	Callback when the styles of an object have changed.

Description

Experimental.UIElements.Slider is a controller allowing the selection of a value from a range (min, max, value).

Inherited Members

Static Properties

defaultFocusIndex	The default focus index for newly created elements.

Properties

value	The value associated with the field.
direction	This is the actual property to contain the direction of the slider.
highValue	This is the maximum value that the slider encodes.
lowValue	This is the minimum value that the slider encodes.
pageSize	This is a generic page size used to change the value when clicking in the slider.
range	This is the range from the minimum value to the maximum value of the slider.
value	The actual value of the slider.
binding	Binding object that will be updated.
bindingPath	Path of the target property to be bound.
canGrabFocus	Return true if the element can be focused.
focusController	Return the focus controller for this element.
focusIndex	An integer used to sort focusables in the focus ring. A negative value means that the element can not be focused.
childCount	Number of child elements in this object's contentContainer
clippingOptions	Should this element clip painting to its boundaries.
contentContainer	child elements are added to this element, usually this
dataWatch	Access to this element data watch interface.
enabledInHierarchy	Returns true if the VisualElement is enabled in its own hierarchy.
enabledSelf	Returns true if the VisualElement is enabled locally.

persistenceKey	Used for view data persistence (ie. tree expanded states, scroll position, zoom level).
schedule	Retrieves this VisualElement's IVisualElementScheduler
shadow	Access to this element physical hierarchy
style	Reference to the style object of this element.
this[int]	Access to this element physical hierarchy
tooltip	Text to display inside an information box after the user hovers the element for a small amount of time.
userData	This property can be used to associate application-specific user data with this VisualElement.

Public Methods

SetValueWithoutNotify	Allow to set a value without being of the change, if any.
AdjustDragElement	Method used to adjust the dragelement. Mainly used in a scroller.
HandleEvent	Handle an event, most often by executing the callbacks associated with the event.
HasBubbleUpHandlers	Return true if event handlers for the event propagation BubbleUp phase have been attached on this object.
HasTrickleDownHandlers	Returns true if event handlers, for the event propagation TrickleDown phase, are attached to this object.
RegisterCallback	Adds an event handler to the instance. If the event handler has already been registered for the same phase (either TrickleDown or BubbleUp) then this method has no effect.
SendEvent	Sends an event to the event handler.
UnregisterCallback	Remove callback from the instance.
Blur	Tell the element to release the focus.
Focus	Attempt to give the focus to this element.
Add	Add an element to this element's contentContainer
AddStyleSheetPath	Adds this stylesheet file to this element list of applied styles
BringToFront	Brings this element to the end of its parent children list. The element will be visually in front of any overlapping sibling elements.
Children	Returns the elements from its contentContainer

Clear	Remove all child elements from this element's contentContainer
Contains	Returns true if the element is a direct child of this VisualElement
ElementAt	Retrieves the child element at position
EnableInClassList	Enables or disables the class with the given name.
FindAncestorUserData	Searchs up the hierachy of this VisualElement and retrieves stored userData, if any is found.
FindCommonAncestor	Finds the lowest commont ancestor between two VisualElements inside the VisualTree hierarchy
GetEnumerator	Allows to iterate into this elements children
GetFirstAncestorOfType	Walks up the hierarchy, starting from this element's parent, and returns the first VisualElement of this type
GetFirstOfType	Walks up the hierarchy, starting from this element, and returns the first VisualElement of this type
GetFullHierarchicalPersistenceKey	Combine this VisualElement's VisualElement.persistenceKey with those of its parents to create a more unique key for use with VisualElement.GetOrCreatePersistentData.
GetOrCreatePersistentData	Takes a reference to an existing persisted object and a key and returns the object either filled with the persisted state or as-is.
HasStyleSheetPath	Checks if this stylesheet file is in this element list of applied styles
IndexOf	Retrieves the child index of the specified VisualElement.
Insert	Insert an element into this element's contentContainer
MarkDirtyRepaint	Triggers a repaint of the VisualElement on the next frame.
OnPersistentDataReady	Called when the persistent data is accessible and/or when the data or persistence key have changed (VisualElement is properly parented).
OverwriteFromPersistedData	Overwrite object from the persistent data store.
PlaceBehind	Places this element right before the sibling element in their parent children list. If the element and the sibling position overlap, the element will be visually behind of its sibling.
PlaceInFront	Places this element right after the sibling element in their parent children list. If the element and the sibling position overlap, the element will be visually in front of its sibling.
Remove	Removes this child from the hierarchy
RemoveAt	Remove the child element located at this position from this element's

	contentContainer
RemoveFromHierarchy	Removes this element from its parent hierarchy
RemoveStyleSheetPath	Removes this stylesheet file from this element list of applied styles
SavePersistentData	Write persistence data to file.
SendEvent	Sends an event to the event handler.
SendToBack	Sends this element to the beginning of its parent children list. The element will be visually behind any overlapping sibling elements.
SetEnabled	Changes whether the current VisualElement is enabled or not. When disabled, a VisualElement does not receive most events.
Sort	Reorders child elements from this VisualElement contentContainer.
ToggleInClassList	Toggles between adding and removing the given class name from the class list.

Protected Methods

OnStyleResolved	Callback when the styles of an object have changed.

Description

Template Container.

Inherited Members

Static Properties

defaultFocusIndex	The default focus index for newly created elements.

Properties

binding	Binding object that will be updated.
bindingPath	Path of the target property to be bound.
canGrabFocus	Return true if the element can be focused.
focusController	Return the focus controller for this element.
focusIndex	An integer used to sort focusables in the focus ring. A negative value means that the element can not be focused.
childCount	Number of child elements in this object's contentContainer
clippingOptions	Should this element clip painting to its boundaries.
contentContainer	child elements are added to this element, usually this
dataWatch	Access to this element data watch interface.
enabledInHierarchy	Returns true if the VisualElement is enabled in its own hierarchy.
enabledSelf	Returns true if the VisualElement is enabled locally.
persistenceKey	Used for view data persistence (ie. tree expanded states, scroll position, zoom level).
schedule	Retrieves this VisualElement's IVisualElementScheduler
shadow	Access to this element physical hierarchy
style	Reference to the style object of this element.
this[int]	Access to this element physical hierarchy
tooltip	Text to display inside an information box after the user hovers the element for a small amount of time.
userData	This property can be used to associate application-specific user data with this

VisualElement.

Public Methods

HandleEvent	Handle an event, most often by executing the callbacks associated with the event.
HasBubbleUpHandlers	Return true if event handlers for the event propagation BubbleUp phase have been attached on this object.
HasTrickleDownHandlers	Returns true if event handlers, for the event propagation TrickleDown phase, are attached to this object.
RegisterCallback	Adds an event handler to the instance. If the event handler has already been registered for the same phase (either TrickleDown or BubbleUp) then this method has no effect.
SendEvent	Sends an event to the event handler.
UnregisterCallback	Remove callback from the instance.
Blur	Tell the element to release the focus.
Focus	Attempt to give the focus to this element.
Add	Add an element to this element's contentContainer
AddStyleSheetPath	Adds this stylesheet file to this element list of applied styles
BringToFront	Brings this element to the end of its parent children list. The element will be visually in front of any overlapping sibling elements.
Children	Returns the elements from its contentContainer
Clear	Remove all child elements from this element's contentContainer
Contains	Returns true if the element is a direct child of this VisualElement
ElementAt	Retrieves the child element at position
EnableInClassList	Enables or disables the class with the given name.
FindAncestorUserData	Searchs up the hierachy of this VisualElement and retrieves stored userData, if any is found.
FindCommonAncestor	Finds the lowest commont ancestor between two VisualElements inside the VisualTree hierarchy
GetEnumerator	Allows to iterate into this elements children
GetFirstAncestorOfType	Walks up the hierarchy, starting from this element's parent, and returns the first VisualElement of this type

GetFirstOfType	Walks up the hierarchy, starting from this element, and returns the first VisualElement of this type
GetFullHierarchicalPersistenceKey	Combine this VisualElement's VisualElement.persistenceKey with those of its parents to create a more unique key for use with VisualElement.GetOrCreatePersistentData.
GetOrCreatePersistentData	Takes a reference to an existing persisted object and a key and returns the object either filled with the persisted state or as-is.
HasStyleSheetPath	Checks if this stylesheet file is in this element list of applied styles
IndexOf	Retrieves the child index of the specified VisualElement.
Insert	Insert an element into this element's contentContainer
MarkDirtyRepaint	Triggers a repaint of the VisualElement on the next frame.
OnPersistentDataReady	Called when the persistent data is accessible and/or when the data or persistence key have changed (VisualElement is properly parented).
OverwriteFromPersistedData	Overwrite object from the persistent data store.
PlaceBehind	Places this element right before the sibling element in their parent children list. If the element and the sibling position overlap, the element will be visually behind of its sibling.
PlaceInFront	Places this element right after the sibling element in their parent children list. If the element and the sibling position overlap, the element will be visually in front of its sibling.
Remove	Removes this child from the hierarchy
RemoveAt	Remove the child element located at this position from this element's contentContainer
RemoveFromHierarchy	Removes this element from its parent hierarchy
RemoveStyleSheetPath	Removes this stylesheet file from this element list of applied styles
SavePersistentData	Write persistence data to file.
SendEvent	Sends an event to the event handler.
SendToBack	Sends this element to the beginning of its parent children list. The element will be visually behind any overlapping sibling elements.
SetEnabled	Changes whether the current VisualElement is enabled or not. When disabled, a VisualElement does not receive most events.
Sort	Reorders child elements from this VisualElement contentContainer.
ToggleInClassList	Toggles between adding and removing the given class name from the

class list.

Protected Methods

OnStyleResolved	Callback when the styles of an object have changed.

Description

A textfield is a rectangular area where the user can edit a string.

Properties

isPasswordField	Set this to true to mask the characters and false if otherwise.
multiline	Set this to true to allow multiple lines in the textfield and false if otherwise.
value	The string currently being exposed by the field.

Constructors

TextField	Creates a new textfield.

Public Methods

OnPersistentDataReady	Called when the persistent data is accessible and/or when the data or persistence key have changed (VisualElement is properly parented).
SelectRange	Selects text in the textfield between cursorIndex and selectionIndex.

Inherited Members

Static Properties

defaultFocusIndex	The default focus index for newly created elements.

Properties

value	The value associated with the field.
binding	Binding object that will be updated.
bindingPath	Path of the target property to be bound.
canGrabFocus	Return true if the element can be focused.
focusController	Return the focus controller for this element.
focusIndex	An integer used to sort focusables in the focus ring. A negative value means that the element can not be focused.

cursorColor	Color of the cursor.
cursorIndex	The current cursor position index in the text input field.
doubleClickSelectsWord	Controls whether double clicking selects the word under the mouse pointer or not.
isDelayed	If set to true, the value property is not updated until either the user presses Enter or the text field loses focus.
isPasswordField	Returns true if the field is used to edit a password.
maskChar	The character used for masking in a password field.
maxLength	Maximum number of characters for the field.
selectIndex	The current selection position index in the text input field.
selectionColor	Background color of selected text.
tripleClickSelectsLine	Controls whether triple clicking selects the entire line under the mouse pointer or not.
childCount	Number of child elements in this object's contentContainer
clippingOptions	Should this element clip painting to its boundaries.
contentContainer	child elements are added to this element, usually this
dataWatch	Access to this element data watch interface.
enabledInHierarchy	Returns true if the VisualElement is enabled in its own hierarchy.
enabledSelf	Returns true if the VisualElement is enabled locally.
persistenceKey	Used for view data persistence (ie. tree expanded states, scroll position, zoom level).
schedule	Retrieves this VisualElement's IVisualElementScheduler
shadow	Access to this element physical hierarchy
style	Reference to the style object of this element.
this[int]	Access to this element physical hierarchy
tooltip	Text to display inside an information box after the user hovers the element for a small amount of time.
userData	This property can be used to associate application-specific user data with this VisualElement.

Public Methods

SetValueWithoutNotify	Allow to set a value without being of the change, if any.
HandleEvent	Handle an event, most often by executing the callbacks associated with the event.
HasBubbleUpHandlers	Return true if event handlers for the event propagation BubbleUp phase have been attached on this object.
HasTrickleDownHandlers	Returns true if event handlers, for the event propagation TrickleDown phase, are attached to this object.
RegisterCallback	Adds an event handler to the instance. If the event handler has already been registered for the same phase (either TrickleDown or BubbleUp) then this method has no effect.
SendEvent	Sends an event to the event handler.
UnregisterCallback	Remove callback from the instance.
Blur	Tell the element to release the focus.
Focus	Attempt to give the focus to this element.
SelectAll	Selects all the text.
Add	Add an element to this element's contentContainer
AddStyleSheetPath	Adds this stylesheet file to this element list of applied styles
BringToFront	Brings this element to the end of its parent children list. The element will be visually in front of any overlapping sibling elements.
Children	Returns the elements from its contentContainer
Clear	Remove all child elements from this element's contentContainer
Contains	Returns true if the element is a direct child of this VisualElement
ElementAt	Retrieves the child element at position
EnableInClassList	Enables or disables the class with the given name.
FindAncestorUserData	Searchs up the hierachy of this VisualElement and retrieves stored userData, if any is found.
FindCommonAncestor	Finds the lowest commont ancestor between two VisualElements inside the VisualTree hierarchy
GetEnumerator	Allows to iterate into this elements children
GetFirstAncestorOfType	Walks up the hierarchy, starting from this element's parent, and returns the first VisualElement of this type
GetFirstOfType	Walks up the hierarchy, starting from this element, and returns the first

	VisualElement of this type
GetFullHierarchicalPersistenceKey	Combine this VisualElement's VisualElement.persistenceKey with those of its parents to create a more unique key for use with VisualElement.GetOrCreatePersistentData.
GetOrCreatePersistentData	Takes a reference to an existing persisted object and a key and returns the object either filled with the persisted state or as-is.
HasStyleSheetPath	Checks if this stylesheet file is in this element list of applied styles
IndexOf	Retrieves the child index of the specified VisualElement.
Insert	Insert an element into this element's contentContainer
MarkDirtyRepaint	Triggers a repaint of the VisualElement on the next frame.
OverwriteFromPersistedData	Overwrite object from the persistent data store.
PlaceBehind	Places this element right before the sibling element in their parent children list. If the element and the sibling position overlap, the element will be visually behind of its sibling.
PlaceInFront	Places this element right after the sibling element in their parent children list. If the element and the sibling position overlap, the element will be visually in front of its sibling.
Remove	Removes this child from the hierarchy
RemoveAt	Remove the child element located at this position from this element's contentContainer
RemoveFromHierarchy	Removes this element from its parent hierarchy
RemoveStyleSheetPath	Removes this stylesheet file from this element list of applied styles
SavePersistentData	Write persistence data to file.
SendEvent	Sends an event to the event handler.
SendToBack	Sends this element to the beginning of its parent children list. The element will be visually behind any overlapping sibling elements.
SetEnabled	Changes whether the current VisualElement is enabled or not. When disabled, a VisualElement does not receive most events.
Sort	Reorders child elements from this VisualElement contentContainer.
ToggleInClassList	Toggles between adding and removing the given class name from the class list.

Protected Methods

BuildContextualMenu	Add menu items to the text field contextual menu.
OnStyleResolved	Callback when the styles of an object have changed.

Implements interfaces:IEventHandler

Description

Abstract base class used for all text-based fields.

Properties

cursorColor	Color of the cursor.
cursorIndex	The current cursor position index in the text input field.
doubleClickSelectsWord	Controls whether double clicking selects the word under the mouse pointer or not.
isDelayed	If set to true, the value property is not updated until either the user presses Enter or the text field loses focus.
isPasswordField	Returns true if the field is used to edit a password.
maskChar	The character used for masking in a password field.
maxLength	Maximum number of characters for the field.
selectIndex	The current selection position index in the text input field.
selectionColor	Background color of selected text.
tripleClickSelectsLine	Controls whether triple clicking selects the entire line under the mouse pointer or not.

Constructors

TextInputFieldBase_1	Constructor.

Public Methods

SelectAll	Selects all the text.

Protected Methods

BuildContextualMenu	Add menu items to the text field contextual menu.

Inherited Members

Static Properties

defaultFocusIndex	The default focus index for newly created elements.

Properties

value	The value associated with the field.
binding	Binding object that will be updated.
bindingPath	Path of the target property to be bound.
canGrabFocus	Return true if the element can be focused.
focusController	Return the focus controller for this element.
focusIndex	An integer used to sort focusables in the focus ring. A negative value means that the element can not be focused.
childCount	Number of child elements in this object's contentContainer
clippingOptions	Should this element clip painting to its boundaries.
contentContainer	child elements are added to this element, usually this
dataWatch	Access to this element data watch interface.
enabledInHierarchy	Returns true if the VisualElement is enabled in its own hierarchy.
enabledSelf	Returns true if the VisualElement is enabled locally.
persistenceKey	Used for view data persistence (ie. tree expanded states, scroll position, zoom level).
schedule	Retrieves this VisualElement's IVisualElementScheduler
shadow	Access to this element physical hierarchy
style	Reference to the style object of this element.
this[int]	Access to this element physical hierarchy
tooltip	Text to display inside an information box after the user hovers the element for a small amount of time.
userData	This property can be used to associate application-specific user data with this VisualElement.

Public Methods

SetValueWithoutNotify	Allow to set a value without being of the change, if any.

HandleEvent	Handle an event, most often by executing the callbacks associated with the event.
HasBubbleUpHandlers	Return true if event handlers for the event propagation BubbleUp phase have been attached on this object.
HasTrickleDownHandlers	Returns true if event handlers, for the event propagation TrickleDown phase, are attached to this object.
RegisterCallback	Adds an event handler to the instance. If the event handler has already been registered for the same phase (either TrickleDown or BubbleUp) then this method has no effect.
SendEvent	Sends an event to the event handler.
UnregisterCallback	Remove callback from the instance.
Blur	Tell the element to release the focus.
Focus	Attempt to give the focus to this element.
Add	Add an element to this element's contentContainer
AddStyleSheetPath	Adds this stylesheet file to this element list of applied styles
BringToFront	Brings this element to the end of its parent children list. The element will be visually in front of any overlapping sibling elements.
Children	Returns the elements from its contentContainer
Clear	Remove all child elements from this element's contentContainer
Contains	Returns true if the element is a direct child of this VisualElement
ElementAt	Retrieves the child element at position
EnableInClassList	Enables or disables the class with the given name.
FindAncestorUserData	Searchs up the hierachy of this VisualElement and retrieves stored userData, if any is found.
FindCommonAncestor	Finds the lowest commont ancestor between two VisualElements inside the VisualTree hierarchy
GetEnumerator	Allows to iterate into this elements children
GetFirstAncestorOfType	Walks up the hierarchy, starting from this element's parent, and returns the first VisualElement of this type
GetFirstOfType	Walks up the hierarchy, starting from this element, and returns the first VisualElement of this type
GetFullHierarchicalPersistenceKey	Combine this VisualElement's VisualElement.persistenceKey with those of

	its parents to create a more unique key for use with VisualElement.GetOrCreatePersistentData.
GetOrCreatePersistentData	Takes a reference to an existing persisted object and a key and returns the object either filled with the persisted state or as-is.
HasStyleSheetPath	Checks if this stylesheet file is in this element list of applied styles
IndexOf	Retrieves the child index of the specified VisualElement.
Insert	Insert an element into this element's contentContainer
MarkDirtyRepaint	Triggers a repaint of the VisualElement on the next frame.
OnPersistentDataReady	Called when the persistent data is accessible and/or when the data or persistence key have changed (VisualElement is properly parented).
OverwriteFromPersistedData	Overwrite object from the persistent data store.
PlaceBehind	Places this element right before the sibling element in their parent children list. If the element and the sibling position overlap, the element will be visually behind of its sibling.
PlaceInFront	Places this element right after the sibling element in their parent children list. If the element and the sibling position overlap, the element will be visually in front of its sibling.
Remove	Removes this child from the hierarchy
RemoveAt	Remove the child element located at this position from this element's contentContainer
RemoveFromHierarchy	Removes this element from its parent hierarchy
RemoveStyleSheetPath	Removes this stylesheet file from this element list of applied styles
SavePersistentData	Write persistence data to file.
SendEvent	Sends an event to the event handler.
SendToBack	Sends this element to the beginning of its parent children list. The element will be visually behind any overlapping sibling elements.
SetEnabled	Changes whether the current VisualElement is enabled or not. When disabled, a VisualElement does not receive most events.
Sort	Reorders child elements from this VisualElement contentContainer.
ToggleInClassList	Toggles between adding and removing the given class name from the class list.

Protected Methods

OnStyleResolved	Callback when the styles of an object have changed.

Description

This is the Toggle field.

Properties

text	Optional text after the toggle.

Inherited Members

Static Properties

defaultFocusIndex	The default focus index for newly created elements.

Properties

value	The value associated with the field.
binding	Binding object that will be updated.
bindingPath	Path of the target property to be bound.
canGrabFocus	Return true if the element can be focused.
focusController	Return the focus controller for this element.
focusIndex	An integer used to sort focusables in the focus ring. A negative value means that the element can not be focused.
childCount	Number of child elements in this object's contentContainer
clippingOptions	Should this element clip painting to its boundaries.
contentContainer	child elements are added to this element, usually this
dataWatch	Access to this element data watch interface.
enabledInHierarchy	Returns true if the VisualElement is enabled in its own hierarchy.
enabledSelf	Returns true if the VisualElement is enabled locally.
persistenceKey	Used for view data persistence (ie. tree expanded states, scroll position, zoom level).
schedule	Retrieves this VisualElement's IVisualElementScheduler
shadow	Access to this element physical hierarchy

style	Reference to the style object of this element.
this[int]	Access to this element physical hierarchy
tooltip	Text to display inside an information box after the user hovers the element for a small amount of time.
userData	This property can be used to associate application-specific user data with this VisualElement.

Public Methods

SetValueWithoutNotify	Allow to set a value without being of the change, if any.
HandleEvent	Handle an event, most often by executing the callbacks associated with the event.
HasBubbleUpHandlers	Return true if event handlers for the event propagation BubbleUp phase have been attached on this object.
HasTrickleDownHandlers	Returns true if event handlers, for the event propagation TrickleDown phase, are attached to this object.
RegisterCallback	Adds an event handler to the instance. If the event handler has already been registered for the same phase (either TrickleDown or BubbleUp) then this method has no effect.
SendEvent	Sends an event to the event handler.
UnregisterCallback	Remove callback from the instance.
Blur	Tell the element to release the focus.
Focus	Attempt to give the focus to this element.
Add	Add an element to this element's contentContainer
AddStyleSheetPath	Adds this stylesheet file to this element list of applied styles
BringToFront	Brings this element to the end of its parent children list. The element will be visually in front of any overlapping sibling elements.
Children	Returns the elements from its contentContainer
Clear	Remove all child elements from this element's contentContainer
Contains	Returns true if the element is a direct child of this VisualElement
ElementAt	Retrieves the child element at position
EnableInClassList	Enables or disables the class with the given name.

FindAncestorUserData	Searchs up the hierachy of this VisualElement and retrieves stored userData, if any is found.
FindCommonAncestor	Finds the lowest commont ancestor between two VisualElements inside the VisualTree hierarchy
GetEnumerator	Allows to iterate into this elements children
GetFirstAncestorOfType	Walks up the hierarchy, starting from this element's parent, and returns the first VisualElement of this type
GetFirstOfType	Walks up the hierarchy, starting from this element, and returns the first VisualElement of this type
GetFullHierarchicalPersistenceKey	Combine this VisualElement's VisualElement.persistenceKey with those of its parents to create a more unique key for use with VisualElement.GetOrCreatePersistentData.
GetOrCreatePersistentData	Takes a reference to an existing persisted object and a key and returns the object either filled with the persisted state or as-is.
HasStyleSheetPath	Checks if this stylesheet file is in this element list of applied styles
IndexOf	Retrieves the child index of the specified VisualElement.
Insert	Insert an element into this element's contentContainer
MarkDirtyRepaint	Triggers a repaint of the VisualElement on the next frame.
OnPersistentDataReady	Called when the persistent data is accessible and/or when the data or persistence key have changed (VisualElement is properly parented).
OverwriteFromPersistedData	Overwrite object from the persistent data store.
PlaceBehind	Places this element right before the sibling element in their parent children list. If the element and the sibling position overlap, the element will be visually behind of its sibling.
PlaceInFront	Places this element right after the sibling element in their parent children list. If the element and the sibling position overlap, the element will be visually in front of its sibling.
Remove	Removes this child from the hierarchy
RemoveAt	Remove the child element located at this position from this element's contentContainer
RemoveFromHierarchy	Removes this element from its parent hierarchy
RemoveStyleSheetPath	Removes this stylesheet file from this element list of applied styles
SavePersistentData	Write persistence data to file.

SendEvent	Sends an event to the event handler.
SendToBack	Sends this element to the beginning of its parent children list. The element will be visually behind any overlapping sibling elements.
SetEnabled	Changes whether the current VisualElement is enabled or not. When disabled, a VisualElement does not receive most events.
Sort	Reorders child elements from this VisualElement contentContainer.
ToggleInClassList	Toggles between adding and removing the given class name from the class list.

Protected Methods

OnStyleResolved	Callback when the styles of an object have changed.

Description

UQuery is a set of extension methods allowing you to select individual or collection of visualElements inside a complex hierarchy.

Description

Utility Object that contructs a set of selection rules to be ran on a root visual element.

Constructors

UQuery.QueryBuilder_1	Initializes a QueryBuilder.

Public Methods

Active	Selects all elements that are active.
AtIndex	Convenience overload, shorthand for Build().AtIndex().
Build	Compiles the selection rules into a QueryState object.
Checked	Selects all elements that are checked.
Children	Selects all direct child elements of elements matching the previous rules.
Class	Selects all elements with the given class. Not to be confused with Type (see OfType<>()).
Descendents	Selects all elements that are descendants of currently matching ancestors.
Enabled	Selects all elements that are enabled.
First	Convenience overload, shorthand for Build().First().
Focused	Selects all elements that are enabled.
ForEach	Convenience overload, shorthand for Build().ForEach().
Hovered	Selects all elements that are hovered.
Last	Convenience overload, shorthand for Build().Last().
Name	Selects element with this name.
NotActive	Selects all elements that are not active.
NotChecked	Selects all elements that npot checked.
NotEnabled	Selects all elements that are not enabled.
NotFocused	Selects all elements that don't currently own the focus.
NotHovered	Selects all elements that are not hovered.
NotSelected	Selects all elements that are not selected.

NotVisible	Selects all elements that are not visible.
OfType	Selects all elements of the specified Type (eg: Label, Button, ScrollView, etc).
Selected	Selects all elements that are selected.
ToList	Convenience method. shorthand for Build().ToList.
Visible	Selects all elements that are visible.
Where	Selects all elements satifying the predicate.

Description

Query object containing all the selection rules. Can be saved and rerun later without re-allocating memory.

Public Methods

AtIndex	Selects the n th element matching all the criteria, or null if not enough elements were found.
First	The first element matching all the criteria, or null if none was found.
ForEach	Invokes function on all elements matching the query.
Last	The last element matching all the criteria, or null if none was found.
RebuildOn	Creates a new QueryState with the same selection rules, applied on another VisualElement.
ToList	Returns a list containing elements satisfying selection rules.

Description

UQuery is a set of extension methods allowing you to select individual or collection of visualElements inside a complex hierarchy.

Static Methods

Q	Convenience overload, shorthand for Query<T>.Build().First().
Query	Initializes a QueryBuilder with the specified selection rules.

Description

Base class for describing an XML attribute.

Properties

defaultValueAsString	The default value for the attribute, as a string.
name	The attribute name.
obsoleteNames	A list of obsolete names for this attribute.
restriction	Restrictions on the possible values of the attribute.
type	Attribute type.
typeNamespace	Attribute namespace.
use	Whether the attribute is optional, required or prohibited.

Protected Methods

GetValueFromBag	Get the attribute value from the attribute bag.

Description

An enum to describe attribute use.

Properties

None	There is no restriction on the use of this attribute with the element.
Optional	The attribute is optional for the element.
Prohibited	The attribute should not appear for the element.
Required	The attribute must appear in the element tag.

Description

Describes a XML `bool` attribute.

Properties

defaultValue	The default value for the attribute.
defaultValueAsString	The default value for the attribute, as a string.

Constructors

UxmlBoolAttributeDescription	Constructor.

Public Methods

GetValueFromBag	Retrieves the value of this attribute from the attribute bag. Returns it if it is found, otherwise return defaultValue.

Inherited Members

Properties

defaultValueAsString	The default value for the attribute, as a string.
name	The attribute name.
obsoleteNames	A list of obsolete names for this attribute.
restriction	Restrictions on the possible values of the attribute.
type	Attribute type.
typeNamespace	Attribute namespace.
use	Whether the attribute is optional, required or prohibited.

Protected Methods

GetValueFromBag	Get the attribute value from the attribute bag.

Description

Describe an allowed child element for an element.

Properties

elementName	The name of the allowed child element.
elementNamespace	The namespace name of the allowed child element.

Constructors

UxmlChildElementDescription	Constructor.

Description

Describes a XML attribute representing a Color as a `string`.

Properties

defaultValue	The default value for the attribute.
defaultValueAsString	The default value for the attribute, as a string.

Constructors

UxmlColorAttributeDescription	Constructor.

Public Methods

GetValueFromBag	Retrieves the value of this attribute from the attribute bag. Returns it if it is found, otherwise return defaultValue.

Inherited Members

Properties

defaultValueAsString	The default value for the attribute, as a string.
name	The attribute name.
obsoleteNames	A list of obsolete names for this attribute.
restriction	Restrictions on the possible values of the attribute.
type	Attribute type.
typeNamespace	Attribute namespace.
use	Whether the attribute is optional, required or prohibited.

Protected Methods

GetValueFromBag	Get the attribute value from the attribute bag.

Description

Describes a XML `double` attribute.

Properties

defaultValue	The default value for the attribute.
defaultValueAsString	The default value for the attribute, as a string.

Constructors

UxmlDoubleAttributeDescription	Constructor.

Public Methods

GetValueFromBag	Retrieves the value of this attribute from the attribute bag. Returns it if it is found, otherwise return defaultValue.

Inherited Members

Properties

defaultValueAsString	The default value for the attribute, as a string.
name	The attribute name.
obsoleteNames	A list of obsolete names for this attribute.
restriction	Restrictions on the possible values of the attribute.
type	Attribute type.
typeNamespace	Attribute namespace.
use	Whether the attribute is optional, required or prohibited.

Protected Methods

GetValueFromBag	Get the attribute value from the attribute bag.

Description

Describes a XML attribute representing an `enum` as a `string`.

Properties

defaultValue	The default value for the attribute.
defaultValueAsString	The default value for the attribute, as a string.

Constructors

UxmlEnumAttributeDescription_1	Constructor.

Public Methods

GetValueFromBag	Retrieves the value of this attribute from the attribute bag. Returns it if it is found, otherwise return defaultValue.

Inherited Members

Properties

defaultValueAsString	The default value for the attribute, as a string.
name	The attribute name.
obsoleteNames	A list of obsolete names for this attribute.
restriction	Restrictions on the possible values of the attribute.
type	Attribute type.
typeNamespace	Attribute namespace.
use	Whether the attribute is optional, required or prohibited.

Protected Methods

GetValueFromBag	Get the attribute value from the attribute bag.

Description

Restricts the value of an attribute to be taken from a list of values.

Properties

values	The list of values the attribute can take.

Constructors

UxmlEnumeration	Constructor.

Public Methods

Equals	Indicates whether the current UxmlEnumeration object is equal to another object of the same type.

Inherited Members

Description

UxmlFactory specialization for classes that derive from VisualElement and that shares its traits, VisualElementTraits.

Constructors

UxmlFactory_1	Constructor.

Inherited Members

Properties

canHaveAnyAttribute	Returns UxmlTraits.canHaveAnyAttribute (where UxmlTraits is the argument for T1).
substituteForTypeName	Returns an empty string if T0 is not VisualElement; otherwise, returns "VisualElement".
substituteForTypeNamespace	Returns the namespace for substituteForTypeName.
substituteForTypeQualifiedName	Returns the fully qualified name for substituteForTypeName.
uxmlAttributesDescription	Returns an empty enumerable.
uxmlChildElementsDescription	Returns an empty enumerable.
uxmlName	Returns the type name of T0.
uxmlNamespace	Returns the namespace name of T0.
uxmlQualifiedName	Returns the typefully qualified name of T0.

Public Methods

AcceptsAttributeBag	Returns true.
Create	Instantiate an object of type T0 and initialize it by calling T1 UxmlTraits.Init method.

Implements interfaces:IUxmlFactory

Description

Generic base class for UXML factories, which instantiate a VisualElement using the data read from a UXML file.

`T0` The type of the element that will be instantiated. It must derive from VisualElement.

`T1` The traits of the element that will be instantiated. It must derive from UxmlTraits.

Properties

canHaveAnyAttribute	Returns UxmlTraits.canHaveAnyAttribute (where UxmlTraits is the argument for T1).
substituteForTypeName	Returns an empty string if T0 is not VisualElement; otherwise, returns "VisualElement".
substituteForTypeNamespace	Returns the namespace for substituteForTypeName.
substituteForTypeQualifiedName	Returns the fully qualified name for substituteForTypeName.
uxmlAttributesDescription	Returns an empty enumerable.
uxmlChildElementsDescription	Returns an empty enumerable.
uxmlName	Returns the type name of T0.
uxmlNamespace	Returns the namespace name of T0.
uxmlQualifiedName	Returns the typefully qualified name of T0.

Public Methods

AcceptsAttributeBag	Returns true.
Create	Instantiate an object of type T0 and initialize it by calling T1 UxmlTraits.Init method.

Description

Describes a XML `float` attribute.

Properties

defaultValue	The default value for the attribute.
defaultValueAsString	The default value for the attribute, as a string.

Constructors

UxmlFloatAttributeDescription	Constructor.

Public Methods

GetValueFromBag	Retrieves the value of this attribute from the attribute bag. Returns it if it is found, otherwise return defaultValue.

Inherited Members

Properties

defaultValueAsString	The default value for the attribute, as a string.
name	The attribute name.
obsoleteNames	A list of obsolete names for this attribute.
restriction	Restrictions on the possible values of the attribute.
type	Attribute type.
typeNamespace	Attribute namespace.
use	Whether the attribute is optional, required or prohibited.

Protected Methods

GetValueFromBag	Get the attribute value from the attribute bag.

Description

Describes a XML `int` attribute.

Properties

defaultValue	The default value for the attribute.
defaultValueAsString	The default value for the attribute, as a string.

Constructors

UxmlIntAttributeDescription	Constructor.

Public Methods

GetValueFromBag	Retrieves the value of this attribute from the attribute bag. Returns it if it is found, otherwise return defaultValue.

Inherited Members

Properties

defaultValueAsString	The default value for the attribute, as a string.
name	The attribute name.
obsoleteNames	A list of obsolete names for this attribute.
restriction	Restrictions on the possible values of the attribute.
type	Attribute type.
typeNamespace	Attribute namespace.
use	Whether the attribute is optional, required or prohibited.

Protected Methods

GetValueFromBag	Get the attribute value from the attribute bag.

Description

Describes a XML `long` attribute.

Properties

defaultValue	The default value for the attribute.
defaultValueAsString	The default value for the attribute, as a string.

Constructors

UxmlLongAttributeDescription	Constructor.

Public Methods

GetValueFromBag	Retrieves the value of this attribute from the attribute bag. Returns it if it is found, otherwise return defaultValue.

Inherited Members

Properties

defaultValueAsString	The default value for the attribute, as a string.
name	The attribute name.
obsoleteNames	A list of obsolete names for this attribute.
restriction	Restrictions on the possible values of the attribute.
type	Attribute type.
typeNamespace	Attribute namespace.
use	Whether the attribute is optional, required or prohibited.

Protected Methods

GetValueFromBag	Get the attribute value from the attribute bag.

Description

Factory for the root `UXML` element.

This factory does not create any VisualElement. It exists solely for the purpose of schema generation.

Properties

substituteForTypeName	Returns the empty string, as the root element can not appear anywhere else bit at the root of the document.
substituteForTypeNamespace	Returns the empty string, as the root element can not appear anywhere else bit at the root of the document.
substituteForTypeQualifiedName	Returns the empty string, as the root element can not appear anywhere else bit at the root of the document.
uxmlName	Returns "UXML".
uxmlQualifiedName	Returns the qualified name for this element.

Constructors

UxmlRootElementFactory	Constructor.

Public Methods

Create	Returns null.

Inherited Members

Properties

canHaveAnyAttribute	Returns UxmlTraits.canHaveAnyAttribute (where UxmlTraits is the argument for T1).
substituteForTypeName	Returns an empty string if T0 is not VisualElement; otherwise, returns "VisualElement".
substituteForTypeNamespace	Returns the namespace for substituteForTypeName.
substituteForTypeQualifiedName	Returns the fully qualified name for substituteForTypeName.
uxmlAttributesDescription	Returns an empty enumerable.

uxmlChildElementsDescription	Returns an empty enumerable.
uxmlName	Returns the type name of T0.
uxmlNamespace	Returns the namespace name of T0.
uxmlQualifiedName	Returns the typefully qualified name of T0.

Public Methods

AcceptsAttributeBag	Returns true.

Description

UxmlTraits for the UXML root element.

Properties

uxmlChildElementsDescription	Returns an enumerable containing UxmlChildElementDescription(typeof(VisualElement)), since the root element can contain VisualElements.

Constructors

UxmlRootElementTraits	Constructor.

Inherited Members

Properties

canHaveAnyAttribute	Must return true if the UXML element attributes are not restricted to the values enumerated by UxmlTraits.uxmlAttributesDescription.
uxmlAttributesDescription	Describes the UXML attributes expected by the element. The attributes enumerated here will appear in the UXML schema.
uxmlChildElementsDescription	Describes the types of element that can appear as children of this element in a UXML file.

Public Methods

Init	Initialize a VisualElement instance with values from the UXML element attributes.

Description

Describes a XML `string` attribute.

Properties

defaultValue	The default value for the attribute.
defaultValueAsString	The default value for the attribute, as a string.

Constructors

UxmlStringAttributeDescription	Constructor.

Public Methods

GetValueFromBag	Retrieves the value of this attribute from the attribute bag. Returns it if it is found, otherwise return defaultValue.

Inherited Members

Properties

defaultValueAsString	The default value for the attribute, as a string.
name	The attribute name.
obsoleteNames	A list of obsolete names for this attribute.
restriction	Restrictions on the possible values of the attribute.
type	Attribute type.
typeNamespace	Attribute namespace.
use	Whether the attribute is optional, required or prohibited.

Protected Methods

GetValueFromBag	Get the attribute value from the attribute bag.

Description

Describes a VisualElement derived class for the parsing of UXML files and the generation of UXML schema definition.

UxmlTraits describes the UXML attributes and children elements of a class deriving from VisualElement. It is used by UxmlFactory to map UXML attributes to the C# class properties when reading UXML documents. It is also used to generate UXML schema definitions.

Properties

canHaveAnyAttribute	Must return true if the UXML element attributes are not restricted to the values enumerated by UxmlTraits.uxmlAttributesDescription.
uxmlAttributesDescription	Describes the UXML attributes expected by the element. The attributes enumerated here will appear in the UXML schema.
uxmlChildElementsDescription	Describes the types of element that can appear as children of this element in a UXML file.

Public Methods

Init	Initialize a VisualElement instance with values from the UXML element attributes.

Description

Base class to restricts the value of an attribute.

Public Methods

Equals	Indicates whether the current UxmlTypeRestriction object is equal to another object of the same type.

Description

Restricts the value of an attribute to be within the specified bounds.

Properties

excludeMax	True if the bounds exclude max.
excludeMin	True if the bounds exclude min.
max	The maximum value for the attribute.
min	The minimum value for the attribute.

Constructors

UxmlValueBounds	Constructor.

Public Methods

Equals	Indicates whether the current UxmlValueBounds object is equal to another object of the same type.

Inherited Members

Description

Restricts the value of an attribute to match a regular expression.

Properties

regex	The regular expression that should be matched by the value.

Constructors

UxmlValueMatches	Constructor.

Public Methods

Equals	Indicates whether the current UxmlValueMatches object is equal to another object of the same type.

Inherited Members

Description

The event sent to probe which elements accepts a command.

Constructors

ValidateCommandEvent	Constructor.

Inherited Members

Properties

commandName	Name of the command.
bubbles	Whether this event type bubbles up in the event propagation path.
currentTarget	The current target of the event. The current target is the element in the propagation path for which event handlers are currently being executed.
dispatch	Whether the event is being dispatched to a visual element. An event cannot be redispatched while it being dispatched. If you need to recursively dispatch an event, it is recommended that you use a copy of the event.
flags	Flags for the event.
imguiEvent	The IMGUIEvent at the source of this event. The source can be null since not all events are generated by IMGUI.
isDefaultPrevented	Return true if the default actions should not be executed for this event.
isImmediatePropagationStopped	Whether StopImmediatePropagation() was called for this event.
isPropagationStopped	Whether StopPropagation() was called for this event.
originalMousePosition	The original mouse position of the IMGUI event, before it is transformed to the current target local coordinates.
pooled	Whether the event is allocated from a pool of events.
propagationPhase	The current propagation phase.
target	The target visual element that received this event. Unlike currentTarget, this target does not change when the event is sent to other elements along the propagation path.
timestamp	The time when the event was created.

tricklesDown	Whether this event is sent down the event propagation path during the TrickleDown phase.

Public Methods

Dispose	Implementation of IDisposable.
GetEventTypeId	Retrieves the type id for this event instance.
PreventDefault	Whether the default actions are prevented from being executed for this event.
StopImmediatePropagation	Immediately stops the propagation of the event. The event is not sent to other elements along the propagation path. This method prevents other event handlers from executing on the current target.
StopPropagation	Stops propagating this event. The event is not sent to other elements along the propagation path. This method does not prevent other event handlers from executing on the current target.
Dispose	Implementation of IDispose.
GetEventTypeId	Retrieves the type id for this event instance.

Protected Methods

Init	Resets the event members to their initial values.
Init	Resets all event members to their initial values.
Init	Resets all event members to their initial values.

Static Methods

GetPooled	Gets an event from the event pool and initializes it with the given values. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().
RegisterEventType	Registers an event class to the event type system.
GetPooled	Gets an event from the event pool. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().
TypeId	Gets the type id for the event class.

Implements interfaces:IStyle, ITransform, IUIElementDataWatch, IVisualElementScheduler

Description

Base class for objects that are part of the UIElements visual tree.

VisualElement contains several features that are common to all controls in UIElements, such as layout, styling and event handling. Several other classes derive from it to implement custom rendering and define behaviour for controls.

Static Properties

defaultFocusIndex	The default focus index for newly created elements.

Properties

childCount	Number of child elements in this object's contentContainer
clippingOptions	Should this element clip painting to its boundaries.
contentContainer	child elements are added to this element, usually this
dataWatch	Access to this element data watch interface.
enabledInHierarchy	Returns true if the VisualElement is enabled in its own hierarchy.
enabledSelf	Returns true if the VisualElement is enabled locally.
persistenceKey	Used for view data persistence (ie. tree expanded states, scroll position, zoom level).
schedule	Retrieves this VisualElement's IVisualElementScheduler
shadow	Access to this element physical hierarchy
style	Reference to the style object of this element.
this[int]	Access to this element physical hierarchy
tooltip	Text to display inside an information box after the user hovers the element for a small amount of time.
userData	This property can be used to associate application-specific user data with this VisualElement.

Public Methods

Add	Add an element to this element's contentContainer

AddStyleSheetPath	Adds this stylesheet file to this element list of applied styles
BringToFront	Brings this element to the end of its parent children list. The element will be visually in front of any overlapping sibling elements.
Children	Returns the elements from its contentContainer
Clear	Remove all child elements from this element's contentContainer
Contains	Returns true if the element is a direct child of this VisualElement
ElementAt	Retrieves the child element at position
EnableInClassList	Enables or disables the class with the given name.
FindAncestorUserData	Searchs up the hierachy of this VisualElement and retrieves stored userData, if any is found.
FindCommonAncestor	Finds the lowest commont ancestor between two VisualElements inside the VisualTree hierarchy
GetEnumerator	Allows to iterate into this elements children
GetFirstAncestorOfType	Walks up the hierarchy, starting from this element's parent, and returns the first VisualElement of this type
GetFirstOfType	Walks up the hierarchy, starting from this element, and returns the first VisualElement of this type
GetFullHierarchicalPersistenceKey	Combine this VisualElement's VisualElement.persistenceKey with those of its parents to create a more unique key for use with VisualElement.GetOrCreatePersistentData.
GetOrCreatePersistentData	Takes a reference to an existing persisted object and a key and returns the object either filled with the persisted state or as-is.
HasStyleSheetPath	Checks if this stylesheet file is in this element list of applied styles
IndexOf	Retrieves the child index of the specified VisualElement.
Insert	Insert an element into this element's contentContainer
MarkDirtyRepaint	Triggers a repaint of the VisualElement on the next frame.
OnPersistentDataReady	Called when the persistent data is accessible and/or when the data or persistence key have changed (VisualElement is properly parented).
OverwriteFromPersistedData	Overwrite object from the persistent data store.
PlaceBehind	Places this element right before the sibling element in their parent children list. If the element and the sibling position overlap, the element will be visually behind of its sibling.

PlaceInFront	Places this element right after the sibling element in their parent children list. If the element and the sibling position overlap, the element will be visually in front of its sibling.
Remove	Removes this child from the hierarchy
RemoveAt	Remove the child element located at this position from this element's contentContainer
RemoveFromHierarchy	Removes this element from its parent hierarchy
RemoveStyleSheetPath	Removes this stylesheet file from this element list of applied styles
SavePersistentData	Write persistence data to file.
SendEvent	Sends an event to the event handler.
SendToBack	Sends this element to the beginning of its parent children list. The element will be visually behind any overlapping sibling elements.
SetEnabled	Changes whether the current VisualElement is enabled or not. When disabled, a VisualElement does not receive most events.
Sort	Reorders child elements from this VisualElement contentContainer.
ToggleInClassList	Toggles between adding and removing the given class name from the class list.

Protected Methods

OnStyleResolved	Callback when the styles of an object have changed.

Inherited Members

Properties

canGrabFocus	Return true if the element can be focused.
focusController	Return the focus controller for this element.
focusIndex	An integer used to sort focusables in the focus ring. A negative value means that the element can not be focused.

Public Methods

HandleEvent	Handle an event, most often by executing the callbacks associated with the event.
HasBubbleUpHandlers	Return true if event handlers for the event propagation BubbleUp phase have

	been attached on this object.
HasTrickleDownHandlers	Returns true if event handlers, for the event propagation TrickleDown phase, are attached to this object.
RegisterCallback	Adds an event handler to the instance. If the event handler has already been registered for the same phase (either TrickleDown or BubbleUp) then this method has no effect.
UnregisterCallback	Remove callback from the instance.
Blur	Tell the element to release the focus.
Focus	Attempt to give the focus to this element.

Description

Hierarchy is a sctuct allowing access to the shadow hierarchy of visual elements

Properties

childCount	Number of child elements in this object's contentContainer
parent	Access the physical parent of this element in the hierarchy
this[int]	Access to this element physical hierarchy

Public Methods

Add	Add an element to this element's contentContainer
Children	Returns the elements from its contentContainer
Clear	Remove all child elements from this element's contentContainer
ElementAt	Retrieves the child element at position
IndexOf	Retrieves the index of the specified VisualElement in the Hierarchy.
Insert	Insert an element into this element's contentContainer
Remove	Removes this child from the hierarchy
RemoveAt	Remove the child element located at this position from this element's contentContainer
Sort	Reorders child elements from this VisualElement contentContainer.

Description

Options to select clipping strategy.

Properties

ClipContents	Will enable clipping. This VisualElement and its children's content will be limited to this element's bounds.
NoClipping	Will disable clipping and let children VisualElements paint outside its bounds.
ClipAndCacheContents	Enables clipping and renders contents to a cache texture.

Description

The modes available to measure VisualElement sizes.

This enum value is passed to UIElements.VisualElement.DoMeasure. This lets UI elements indicate their natural size during the layout algorithm.

See Also:UIElements.VisualElement.MeasureTextSize.

Properties

Undefined	The element should give its preferred width/height without any constraint.
Exactly	The element should give the width/height that is passed in and derive the opposite site from this value (for example, calculate text size from a fixed width).
AtMost	At Most. The element should give its preferred width/height but no more than the value passed.

Description

VisualElementExtensions is a set of extension methods useful for VisualElement.

Static Methods

AddManipulator	Add a manipulator associated to a VisualElement.
RemoveManipulator	Remove a manipulator associated to a VisualElement.
StretchToParentWidth	The given VisualElement's left and right edges will be aligned with the corresponding edges of the parent element.

Description

Define focus change directions for the VisualElementFocusRing.

Static Properties

lastValue	Last value for the direction defined by this class.
left	The focus is moving to the left.
right	The focus is moving to the right.

Inherited Members

Static Properties

lastValue	Last value for the direction defined by this class.
none	The null direction. This is usually used when the focus stays on the same element.
unspecified	Focus came from an unspecified direction, for example after a mouse down.

Operators

int	The underlying integer value for this direction.

Implements interfaces:IFocusRing

Description

Implementation of a linear focus ring. Elements are sorted according to their focusIndex.

Properties

defaultFocusOrder	The focus order for elements having 0 has a focusIndex.

Constructors

VisualElementFocusRing	Constructor.

Public Methods

GetFocusChangeDirection	Get the direction of the focus change for the given event. For example, when the Tab key is pressed, focus should be given to the element to the right in the focus ring.
GetNextFocusable	Get the next element in the given direction.

Description

Ordering of elements in the focus ring.

Properties

ChildOrder	Order elements using a depth-first pre-order traversal of the element tree.
PositionXY	Order elements according to their position, first by X, then by Y.
PositionYX	Order elements according to their position, first by Y, then by X.

Description

Mouse wheel event.

Properties

delta	The amount of scrolling applied with the mouse wheel.

Constructors

WheelEvent	Constructor. Use GetPooled() to get an event from a pool of reusable events.

Protected Methods

Init	Resets the event members to their initial values.

Static Methods

GetPooled	Gets an event from the event pool and initializes it with the given values. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().

Inherited Members

Properties

bubbles	Whether this event type bubbles up in the event propagation path.
currentTarget	The current target of the event. The current target is the element in the propagation path for which event handlers are currently being executed.
dispatch	Whether the event is being dispatched to a visual element. An event cannot be redispatched while it being dispatched. If you need to recursively dispatch an event, it is recommended that you use a copy of the event.
flags	Flags for the event.
imguiEvent	The IMGUIEvent at the source of this event. The source can be null since not all events are generated by IMGUI.
isDefaultPrevented	Return true if the default actions should not be executed for this event.

isImmediatePropagationStopped	Whether StopImmediatePropagation() was called for this event.
isPropagationStopped	Whether StopPropagation() was called for this event.
originalMousePosition	The original mouse position of the IMGUI event, before it is transformed to the current target local coordinates.
pooled	Whether the event is allocated from a pool of events.
propagationPhase	The current propagation phase.
target	The target visual element that received this event. Unlike currentTarget, this target does not change when the event is sent to other elements along the propagation path.
timestamp	The time when the event was created.
tricklesDown	Whether this event is sent down the event propagation path during the TrickleDown phase.
actionKey	Returns true if the platform specific action key is pressed. This key is Command on macOS and Control otherwise.
altKey	Returns true if the Alt key is pressed.
button	Integer representing the pressed mouse button: 0 is left, 1 is right, 2 is center.
clickCount	Number of clicks.
commandKey	Returns true if the Windows/Command key is pressed.
ctrlKey	Returns true if the Control key is pressed.
currentTarget	The current target of the event. The current target is the element in the propagation path for which event handlers are currently being executed.
localMousePosition	The mouse position in the current target coordinate system.
modifiers	Flags holding the pressed modifier keys (Alt, Control, Shift, Windows/Command).
mouseDelta	The difference of the mouse position between the previous mouse event and the current mouse event.
mousePosition	The mouse position in the screen coordinate system.
shiftKey	Returns true if the Shift key is pressed.

Public Methods

Dispose	Implementation of IDisposable.

GetEventTypeId	Retrieves the type id for this event instance.
PreventDefault	Whether the default actions are prevented from being executed for this event.
StopImmediatePropagation	Immediately stops the propagation of the event. The event is not sent to other elements along the propagation path. This method prevents other event handlers from executing on the current target.
StopPropagation	Stops propagating this event. The event is not sent to other elements along the propagation path. This method does not prevent other event handlers from executing on the current target.
Dispose	Implementation of IDispose.
GetEventTypeId	Retrieves the type id for this event instance.

Static Methods

RegisterEventType	Registers an event class to the event type system.
GetPooled	Gets an event from the event pool. Use this function instead of creating new events. Events obtained from this method should be released back to the pool using Dispose().
TypeId	Gets the type id for the event class.

Description

Base interface for ChangeEvent.

Description

Interface for Command events.

Properties

commandName	Name of the command.

Description

Interface for drag and drop events.

Description

Interface for class capable of handling events.

Public Methods

HandleEvent	Handle an event.
HasBubbleUpHandlers	Return true if event handlers for the event propagation BubbleUp phase have been attached on this object.
HasTrickleDownHandlers	Returns true if event handlers, for the event propagation TrickleDown phase, are attached to this object.
SendEvent	Sends an event to the event handler.

Description

Interface for focus events.

Properties

direction	Direction of the focus change.
relatedTarget	Related target. See implementation for specific meaning.

Description

Interface for classes implementing focus rings.

Public Methods

GetFocusChangeDirection	Get the direction of the focus change for the given event. For example, when the Tab key is pressed, focus should be given to the element to the right.
GetNextFocusable	Get the next element in the given direction.

Description

Interface for keyboard events.

Properties

actionKey	Returns true if the platform specific action key is pressed. This key is Command on macOS and Control otherwise.
altKey	Return true if the Alt key is pressed.
character	The character.
commandKey	Return true if the Windows/Command key is pressed.
ctrlKey	Return true if the Control key is pressed.
keyCode	The key code.
modifiers	Flag set holding the pressed modifier keys (Alt, Control, Shift, Windows/Command).
shiftKey	Return true if the Shift key is pressed.

Description

Interface for mouse capture events.

Description

Interface for mouse events.

Properties

actionKey	Returns true if the platform specific action key is pressed. This key is Command on macOS and Control otherwise.
altKey	Return true if the Alt key is pressed.
button	Integer representing the pressed mouse button: 0 is left, 1 is right, 2 is center.
clickCount	Number of clicks.
commandKey	Return true if the Windows/Command key is pressed.
ctrlKey	Return true if the Control key is pressed.
localMousePosition	The mouse position in the current target coordinate system.
modifiers	Flag set holding the pressed modifier keys (Alt, Control, Shift, Windows/Command).
mouseDelta	Mouse position difference between the last mouse event and this one.
mousePosition	The mouse position in the panel coordinate system.
shiftKey	Return true if the Shift key is pressed.

Description

Interface for controls that hold a value and can notify when it is changed by user input.

Properties

value	The Value held by the control.

Public Methods

OnValueChanged	Registers this callback to receive ChangeEvent<T> when value is changed by user input.
RemoveOnValueChanged	Unregisters this callback from receiving ChangeEvent<T> when value is changed by user input.
SetValueWithoutNotify	Set the value and, even if different, does not notify registers callbacks with a ChangeEvent<T>

Description

Interface for classes implementing UI panels.

Properties

focusController	Return the focus controller for this panel.

Description

A reference to a scheduled action.

Description

A scheduler allows you to register actions to be executed at a later point.

Public Methods

Schedule	Add this item to the list of scheduled tasks.
ScheduleForDuration	Schedule this action to be executed later. The item will be automatically unscheduled after it has ran for the amount of time specified with the durationMs parameter.
ScheduleOnce	Schedule this action to be executed later. After the execution, the item will be automatically unscheduled.
ScheduleUntil	Schedule this action to be executed later. Item will be unscheduled when condition is met.
Unschedule	Manually unschedules a previously scheduled action.

Description

This interface provides access to a VisualElement style data.

Reading properties from this object will read from the shared copy of style data, if multiple elements match the same USS rules. When writing to any of the properties the target element will start owning its own style data and stop using the shared copy. Writing to a property will mask the value coming from USS with the provided value however other properties will still match the values from USS.

Properties

alignContent	Alignment of the whole area of children on the cross axis if they span over multiple lines in this container.
alignItems	Alignment of children on the cross axis of this container.
alignSelf	Similar to align-items, but only for this specific element.
backgroundColor	Background color to paint in the element's box.
backgroundImage	Background image to paint in the element's box.
backgroundScaleMode	Background image scaling in the element's box.
borderBottomLeftRadius	This is the radius of the bottom-left corner when a rounded rectangle is drawn in the element's box.
borderBottomRightRadius	This is the radius of the bottom-right corner when a rounded rectangle is drawn in the element's box.
borderBottomWidth	Space reserved for the bottom edge of the border during the layout phase.
borderColor	Color of the border to paint inside the element's box.
borderLeftWidth	Space reserved for the left edge of the border during the layout phase.
borderRadius	This is the radius of every corner when a rounded rectangle is drawn in the element's box.
borderRightWidth	Space reserved for the right edge of the border during the layout phase.
borderTopLeftRadius	This is the radius of the top-left corner when a rounded rectangle is drawn in the element's box.
borderTopRightRadius	This is the radius of the top-right corner when a rounded rectangle is drawn in the element's box.
borderTopWidth	Space reserved for the top edge of the border during the layout phase.

color	Color to use when drawing the text of an element.
cursor	Mouse cursor to display when the mouse pointer is over an element.
flex	Ration of this element in its parent during the layout phase.
flexBasis	Initial main size of a flex item, on the main flex axis. The final layout mught be smaller or larger, according to the flex shrinking and growing determined by the flex property.
flexDirection	Direction of the main axis to layout children in a container.
flexGrow	Specifies how much the item will grow relative to the rest of the flexible items inside the same container.
flexShrink	Specifies how the item will shrink relative to the rest of the flexible items inside the same container.
flexWrap	Placement of children over multiple lines if not enough space is available in this container.
font	Font to draw the element's text.
fontSize	Font size to draw the element's text.
fontStyleAndWeight	Font style and weight (normal, bold, italic) to draw the element's text.
height	Fixed height of an element for the layout.
justifyContent	Justification of children on the main axis of this container.
marginBottom	Space reserved for the bottom edge of the margin during the layout phase.
marginLeft	Space reserved for the left edge of the margin during the layout phase.
marginRight	Space reserved for the right edge of the margin during the layout phase.
marginTop	Space reserved for the top edge of the margin during the layout phase.
maxHeight	Maximum height for an element, when it is flexible or measures its own size.
maxWidth	Maximum width for an element, when it is flexible or measures its own size.
minHeight	Minimum height for an element, when it is flexible or measures its own size.
minWidth	Minimum height for an element, when it is flexible or measures its own size.
paddingBottom	Space reserved for the bottom edge of the padding during the layout phase.
paddingLeft	Space reserved for the left edge of the padding during the layout phase.
paddingRight	Space reserved for the right edge of the padding during the layout phase.
paddingTop	Space reserved for the top edge of the padding during the layout phase.

positionBottom	Bottom distance from the element's box during layout.
positionLeft	Left distance from the element's box during layout.
positionRight	Right distance from the element's box during layout.
positionTop	Top distance from the element's box during layout.
positionType	Element's positioning in its parent container.
sliceBottom	Size of the 9-slice's bottom edge when painting an element's background image.
sliceLeft	Size of the 9-slice's left edge when painting an element's background image.
sliceRight	Size of the 9-slice's right edge when painting an element's background image.
sliceTop	Size of the 9-slice's top edge when painting an element's background image.
textClipping	Clipping if the text does not fit in the element's box.
unityTextAlign	Horizontal and vertical text alignment in the element's box.
visibility	Specifies whether or not an element is visible.
width	Fixed width of an element for the layout.
wordWrap	Word wrapping over multiple lines if not enough space is available to draw the text of an element.

Description

This interface provides access to a VisualElement transform data.

Reading properties of this object gives information about the tranform of its VisualElement. It is possible to write the position, scale and rotation of the transform.

Properties

matrix	Transformation matrix calculated from the position, rotation and scale of the transform (Read Only).
position	The position of the VisualElement's transform.
rotation	The rotation of the VisualElement's transform stored as a Quaternion.
scale	The scale of the VisualElement's transform.

Description

Interface allowing access to this elements datawatch.

Public Methods

RegisterWatch	Starts watching an object. When watched, all changes on an object will trigger the callback to be invoked.
UnregisterWatch	Unregisters a previously watched request.

Description

An internal reference to a data watch request.

Description

This type allows UXML attribute value retrieval during the VisualElement instantiation. An instance will be provided to the factory method - see UXMLFactoryAttribute.

Public Methods

TryGetAttributeValue	Get the value of an attribute as a string.

Description

Interface for UXML factories. While it is not strictly required, concrete factories should derive from the generic class UxmlFactory.

Properties

canHaveAnyAttribute	Must return true if the UXML element attributes are not restricted to the values enumerated by uxmlAttributesDescription.
substituteForTypeName	The type of element for which this element type can substitute for.
substituteForTypeNamespace	The UXML namespace for the type returned by substituteForTypeName.
substituteForTypeQualifiedName	The fully qualified XML name for the type returned by substituteForTypeName.
uxmlAttributesDescription	Describes the UXML attributes expected by the element. The attributes enumerated here will appear in the UXML schema.
uxmlChildElementsDescription	Describes the types of element that can appear as children of this element in a UXML file.
uxmlName	The name of the UXML element read by the factory.
uxmlNamespace	The namespace of the UXML element read by the factory.
uxmlQualifiedName	The fully qualified name of the UXML element read by the factory.

Public Methods

AcceptsAttributeBag	Returns true if the factory accepts the content of the attribute bag.
Create	Instanciate and initialize an object of type T0.

Description

Represents a scheduled task created with a VisualElement's schedule interface.

Properties

element	Returns the VisualElement this object is associated with.
isActive	Will be true when this item is scheduled. Note that an item's callback will only be executed when it's VisualElement is attached to a panel.

Public Methods

Every	Repeats this action after a specified time.
ExecuteLater	Cancels any previously scheduled execution of this item and re-schedules the item.
ForDuration	After specified duration, the item will be automatically unscheduled.
Pause	Removes this item from its VisualElement's scheduler.
Resume	If not already active, will schedule this item on its VisualElement's scheduler.
StartingIn	Adds a delay to the first invokation.
Until	Item will be unscheduled automatically when specified condition is met.

Description

A scheduler allows you to register actions to be executed at a later point.

Public Methods

Execute	Schedule this action to be executed later.

Description

Enum which describes the various types of changes that can occur on a VisualElement.

Properties

PersistentData	Persistence key or parent has changed on the current VisualElement.
PersistentDataPath	Persistence key or parent has changed on some child of the current VisualElement.
All	All change types have been flagged.

Description

The propagation phases of an event.

When an element receives an event, the event is propagated along a path built from the elements between the panel root and the target element. In the TrickleDown phase, the event is sent from the root element to the target parent element. In the AtTarget phase, the event is sent to the target element. In the BubbleUp phase, the event is sent from the target parent element to the root element. In the last phase, the DefaultAction phase, the event is resent to the target element.

Properties

None	The event is not being propagated.
TrickleDown	The event is being sent from the root element to the target parent element.
AtTarget	The event is being sent to the event target.
BubbleUp	The event is being sent to the event target parent element up to the root element.
DefaultAction	The event is being sent to the target element for it to execute its default actions for this event. Event handlers do not get the events in this phase. Instead, ExecuteDefaultAction is called on the target.

Description

Controls how many items can be selected at once.

Properties

None	Selections are disabled.
Single	Only one item is selectable.
Multiple	Multiple items are selectable at once.

Implements interfaces:IPlayable

Description

An implementation of IPlayable that controls playback of a VideoClip.

NOTE: You can use PlayableExtensions methods with VideoClipPlayable objects.

Static Methods

Create	Creates a VideoClipPlayable in the PlayableGraph.

Description

Extension methods for the VideoPlayer class.

Static Methods

GetAudioSampleProvider	Return the AudioSampleProvider for the specified track, used to receive audio samples during playback.

Description

The BaseArmModel class is the base class intended for building "Arm Models" for 3 degrees of freedom (3DOF) controllers. An arm model is intended to approximate a 6 degrees of freedom input device using only the rotation of the 3DOF controller input. This is typically achieved by simulating the arm's motion using the rotation, and acceleration information provided by the device.

Properties

headPoseSource	The head pose that will be used as an input to the arm simulation.
poseSource	The poseSource holds the input source to be used for the 3 degrees of freedom controller data. It is this input that will be translated to a full 6 DOF pose by the Arm Model implementation.

Protected Methods

TryGetTrackingDataFromSource	This method is used to allow BaseArmModel specializations to retrieve data from the XRNode system.

Inherited Members

Properties

enabled	Enabled Behaviours are Updated, disabled Behaviours are not.
isActiveAndEnabled	Has the Behaviour had active and enabled called?
gameObject	The game object this component is attached to. A component is always attached to a game object.
tag	The tag of this game object.
transform	The Transform attached to this GameObject.
runInEditMode	Allow a specific instance of a MonoBehaviour to run in edit mode (only available in the editor).
useGUILayout	Disabling this lets you skip the GUI layout phase.
hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.

Public Methods

TryGetPoseFromProvider	This method is the interface point between PoseProviders and the TrackedPoseDriver. Specializing types will implement this method to provide pose data back to the attached TrackedPoseDriver. When a PoseProvider is attached to a TrackedPoseDriver, the TrackedPoseDriver will call this interface each time the TrackedPoseDriver requests a pose.
BroadcastMessage	Calls the method named methodName on every MonoBehaviour in this game object or any of its children.
CompareTag	Is this game object tagged with tag ?
GetComponent	Returns the component of Type type if the game object has one attached, null if it doesn't.
GetComponentInChildren	Returns the component of Type type in the GameObject or any of its children using depth first search.
GetComponentInParent	Returns the component of Type type in the GameObject or any of its parents.
GetComponents	Returns all components of Type type in the GameObject.
GetComponentsInChildren	Returns all components of Type type in the GameObject or any of its children.
GetComponentsInParent	Returns all components of Type type in the GameObject or any of its parents.
SendMessage	Calls the method named methodName on every MonoBehaviour in this game object.
SendMessageUpwards	Calls the method named methodName on every MonoBehaviour in this game object and on every ancestor of the behaviour.
CancelInvoke	Cancels all Invoke calls on this MonoBehaviour.
Invoke	Invokes the method methodName in time seconds.
InvokeRepeating	Invokes the method methodName in time seconds, then repeatedly every repeatRate seconds.
IsInvoking	Is any invoke on methodName pending?
StartCoroutine	Starts a coroutine.
StopAllCoroutines	Stops all coroutines running on this behaviour.
StopCoroutine	Stops the first coroutine named methodName, or the coroutine stored in routine running on this behaviour.
GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.

Static Methods

print	Logs message to the Unity Console (identical to Debug.Log).
Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Messages

Awake	Awake is called when the script instance is being loaded.
FixedUpdate	This function is called every fixed framerate frame, if the MonoBehaviour is enabled.
LateUpdate	LateUpdate is called every frame, if the Behaviour is enabled.
OnAnimatorIK	Callback for setting up animation IK (inverse kinematics).
OnAnimatorMove	Callback for processing animation movements for modifying root motion.
OnApplicationFocus	Sent to all GameObjects when the player gets or loses focus.
OnApplicationPause	Sent to all GameObjects when the application pauses.
OnApplicationQuit	Sent to all game objects before the application quits.
OnAudioFilterRead	If OnAudioFilterRead is implemented, Unity will insert a custom filter into the audio DSP chain.
OnBecameInvisible	OnBecameInvisible is called when the renderer is no longer visible by any camera.

OnBecameVisible	OnBecameVisible is called when the renderer became visible by any camera.
OnCollisionEnter	OnCollisionEnter is called when this collider/rigidbody has begun touching another rigidbody/collider.
OnCollisionEnter2D	Sent when an incoming collider makes contact with this object's collider (2D physics only).
OnCollisionExit	OnCollisionExit is called when this collider/rigidbody has stopped touching another rigidbody/collider.
OnCollisionExit2D	Sent when a collider on another object stops touching this object's collider (2D physics only).
OnCollisionStay	:ref::OnCollisionStay is called once per frame for every collider/rigidbody that is touching rigidbody/collider.
OnCollisionStay2D	Sent each frame where a collider on another object is touching this object's collider (2D physics only).
OnConnectedToServer	Called on the client when you have successfully connected to a server.
OnControllerColliderHit	OnControllerColliderHit is called when the controller hits a collider while performing a Move.
OnDestroy	Destroying the attached Behaviour will result in the game or Scene receiving OnDestroy.
OnDisable	This function is called when the behaviour becomes disabled.
OnDisconnectedFromServer	Called on the client when the connection was lost or you disconnected from the server.
OnDrawGizmos	Implement OnDrawGizmos if you want to draw gizmos that are also pickable and always drawn.
OnDrawGizmosSelected	Implement OnDrawGizmosSelected to draw a gizmo if the object is selected.
OnEnable	This function is called when the object becomes enabled and active.
OnFailedToConnect	Called on the client when a connection attempt fails for some reason.
OnFailedToConnectToMasterServer	Called on clients or servers when there is a problem connecting to the MasterServer.
OnGUI	OnGUI is called for rendering and handling GUI events.
OnJointBreak	Called when a joint attached to the same game object broke.
OnJointBreak2D	Called when a Joint2D attached to the same game object breaks.

OnMasterServerEvent	Called on clients or servers when reporting events from the MasterServer.
OnMouseDown	OnMouseDown is called when the user has pressed the mouse button while over the GUIElement or Collider.
OnMouseDrag	OnMouseDrag is called when the user has clicked on a GUIElement or Collider and is still holding down the mouse.
OnMouseEnter	Called when the mouse enters the GUIElement or Collider.
OnMouseExit	Called when the mouse is not any longer over the GUIElement or Collider.
OnMouseOver	Called every frame while the mouse is over the GUIElement or Collider.
OnMouseUp	OnMouseUp is called when the user has released the mouse button.
OnMouseUpAsButton	OnMouseUpAsButton is only called when the mouse is released over the same GUIElement or Collider as it was pressed.
OnNetworkInstantiate	Called on objects which have been network instantiated with Network.Instantiate.
OnParticleCollision	OnParticleCollision is called when a particle hits a Collider.
OnParticleSystemStopped	OnParticleSystemStopped is called when all particles in the system have died, and no new particles will be born. New particles cease to be created either after Stop is called, or when the duration property of a non-looping system has been exceeded.
OnParticleTrigger	OnParticleTrigger is called when any particles in a particle system meet the conditions in the trigger module.
OnPlayerConnected	Called on the server whenever a new player has successfully connected.
OnPlayerDisconnected	Called on the server whenever a player disconnected from the server.
OnPostRender	OnPostRender is called after a camera finished rendering the Scene.
OnPreCull	OnPreCull is called before a camera culls the Scene.
OnPreRender	OnPreRender is called before a camera starts rendering the Scene.
OnRenderImage	OnRenderImage is called after all rendering is complete to render image.
OnRenderObject	OnRenderObject is called after camera has rendered the Scene.
OnSerializeNetworkView	Used to customize synchronization of variables in a script watched by a network view.
OnServerInitialized	Called on the server whenever a Network.InitializeServer was invoked

	and has completed.
OnTransformChildrenChanged	This function is called when the list of children of the transform of the GameObject has changed.
OnTransformParentChanged	This function is called when the parent property of the transform of the GameObject has changed.
OnTriggerEnter	OnTriggerEnter is called when the Collider other enters the trigger.
OnTriggerEnter2D	Sent when another object enters a trigger collider attached to this object (2D physics only).
OnTriggerExit	OnTriggerExit is called when the Collider other has stopped touching the trigger.
OnTriggerExit2D	Sent when another object leaves a trigger collider attached to this object (2D physics only).
OnTriggerStay	OnTriggerStay is called once per physics update for every Collider other that is touching the trigger.
OnTriggerStay2D	Sent each frame where another object is within a trigger collider attached to this object (2D physics only).
OnValidate	This function is called when the script is loaded or a value is changed in the inspector (Called in the editor only).
OnWillRenderObject	OnWillRenderObject is called for each camera if the object is visible and not a UI element.
Reset	Reset to default values.
Start	Start is called on the frame when a script is enabled just before any of the Update methods are called the first time.
Update	Update is called every frame, if the MonoBehaviour is enabled.

Description

The BasePoseProvider class is a basic implementation of a PoseProvider. A PoseProvider is used to extend the behavior of the TrackedPoseDriver beyond the devices and endpoints that it can normally track. The PoseProvider will override the values normally used by the TrackedPoseDriver. One example use for the PoseProvider is to allow arm models to be attached to the TrackedPoseDriver.

Public Methods

TryGetPoseFromProvider	This method is the interface point between PoseProviders and the TrackedPoseDriver. Specializing types will implement this method to provide pose data back to the attached TrackedPoseDriver. When a PoseProvider is attached to a TrackedPoseDriver, the TrackedPoseDriver will call this interface each time the TrackedPoseDriver requests a pose.

Inherited Members

Properties

enabled	Enabled Behaviours are Updated, disabled Behaviours are not.
isActiveAndEnabled	Has the Behaviour had active and enabled called?
gameObject	The game object this component is attached to. A component is always attached to a game object.
tag	The tag of this game object.
transform	The Transform attached to this GameObject.
runInEditMode	Allow a specific instance of a MonoBehaviour to run in edit mode (only available in the editor).
useGUILayout	Disabling this lets you skip the GUI layout phase.
hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.

Public Methods

BroadcastMessage	Calls the method named methodName on every MonoBehaviour in this game object or any of its children.
CompareTag	Is this game object tagged with tag ?

GetComponent	Returns the component of Type type if the game object has one attached, null if it doesn't.
GetComponentInChildren	Returns the component of Type type in the GameObject or any of its children using depth first search.
GetComponentInParent	Returns the component of Type type in the GameObject or any of its parents.
GetComponents	Returns all components of Type type in the GameObject.
GetComponentsInChildren	Returns all components of Type type in the GameObject or any of its children.
GetComponentsInParent	Returns all components of Type type in the GameObject or any of its parents.
SendMessage	Calls the method named methodName on every MonoBehaviour in this game object.
SendMessageUpwards	Calls the method named methodName on every MonoBehaviour in this game object and on every ancestor of the behaviour.
CancelInvoke	Cancels all Invoke calls on this MonoBehaviour.
Invoke	Invokes the method methodName in time seconds.
InvokeRepeating	Invokes the method methodName in time seconds, then repeatedly every repeatRate seconds.
IsInvoking	Is any invoke on methodName pending?
StartCoroutine	Starts a coroutine.
StopAllCoroutines	Stops all coroutines running on this behaviour.
StopCoroutine	Stops the first coroutine named methodName, or the coroutine stored in routine running on this behaviour.
GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.

Static Methods

print	Logs message to the Unity Console (identical to Debug.Log).
Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.

FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Messages

Awake	Awake is called when the script instance is being loaded.
FixedUpdate	This function is called every fixed framerate frame, if the MonoBehaviour is enabled.
LateUpdate	LateUpdate is called every frame, if the Behaviour is enabled.
OnAnimatorIK	Callback for setting up animation IK (inverse kinematics).
OnAnimatorMove	Callback for processing animation movements for modifying root motion.
OnApplicationFocus	Sent to all GameObjects when the player gets or loses focus.
OnApplicationPause	Sent to all GameObjects when the application pauses.
OnApplicationQuit	Sent to all game objects before the application quits.
OnAudioFilterRead	If OnAudioFilterRead is implemented, Unity will insert a custom filter into the audio DSP chain.
OnBecameInvisible	OnBecameInvisible is called when the renderer is no longer visible by any camera.
OnBecameVisible	OnBecameVisible is called when the renderer became visible by any camera.
OnCollisionEnter	OnCollisionEnter is called when this collider/rigidbody has begun touching another rigidbody/collider.
OnCollisionEnter2D	Sent when an incoming collider makes contact with this object's collider (2D physics only).
OnCollisionExit	OnCollisionExit is called when this collider/rigidbody has stopped touching another rigidbody/collider.
OnCollisionExit2D	Sent when a collider on another object stops touching this object's

	collider (2D physics only).
OnCollisionStay	:ref::OnCollisionStay is called once per frame for every collider/rigidbody that is touching rigidbody/collider.
OnCollisionStay2D	Sent each frame where a collider on another object is touching this object's collider (2D physics only).
OnConnectedToServer	Called on the client when you have successfully connected to a server.
OnControllerColliderHit	OnControllerColliderHit is called when the controller hits a collider while performing a Move.
OnDestroy	Destroying the attached Behaviour will result in the game or Scene receiving OnDestroy.
OnDisable	This function is called when the behaviour becomes disabled.
OnDisconnectedFromServer	Called on the client when the connection was lost or you disconnected from the server.
OnDrawGizmos	Implement OnDrawGizmos if you want to draw gizmos that are also pickable and always drawn.
OnDrawGizmosSelected	Implement OnDrawGizmosSelected to draw a gizmo if the object is selected.
OnEnable	This function is called when the object becomes enabled and active.
OnFailedToConnect	Called on the client when a connection attempt fails for some reason.
OnFailedToConnectToMasterServer	Called on clients or servers when there is a problem connecting to the MasterServer.
OnGUI	OnGUI is called for rendering and handling GUI events.
OnJointBreak	Called when a joint attached to the same game object broke.
OnJointBreak2D	Called when a Joint2D attached to the same game object breaks.
OnMasterServerEvent	Called on clients or servers when reporting events from the MasterServer.
OnMouseDown	OnMouseDown is called when the user has pressed the mouse button while over the GUIElement or Collider.
OnMouseDrag	OnMouseDrag is called when the user has clicked on a GUIElement or Collider and is still holding down the mouse.
OnMouseEnter	Called when the mouse enters the GUIElement or Collider.
OnMouseExit	Called when the mouse is not any longer over the GUIElement or Collider.

OnMouseOver	Called every frame while the mouse is over the GUIElement or Collider.
OnMouseUp	OnMouseUp is called when the user has released the mouse button.
OnMouseUpAsButton	OnMouseUpAsButton is only called when the mouse is released over the same GUIElement or Collider as it was pressed.
OnNetworkInstantiate	Called on objects which have been network instantiated with Network.Instantiate.
OnParticleCollision	OnParticleCollision is called when a particle hits a Collider.
OnParticleSystemStopped	OnParticleSystemStopped is called when all particles in the system have died, and no new particles will be born. New particles cease to be created either after Stop is called, or when the duration property of a non-looping system has been exceeded.
OnParticleTrigger	OnParticleTrigger is called when any particles in a particle system meet the conditions in the trigger module.
OnPlayerConnected	Called on the server whenever a new player has successfully connected.
OnPlayerDisconnected	Called on the server whenever a player disconnected from the server.
OnPostRender	OnPostRender is called after a camera finished rendering the Scene.
OnPreCull	OnPreCull is called before a camera culls the Scene.
OnPreRender	OnPreRender is called before a camera starts rendering the Scene.
OnRenderImage	OnRenderImage is called after all rendering is complete to render image.
OnRenderObject	OnRenderObject is called after camera has rendered the Scene.
OnSerializeNetworkView	Used to customize synchronization of variables in a script watched by a network view.
OnServerInitialized	Called on the server whenever a Network.InitializeServer was invoked and has completed.
OnTransformChildrenChanged	This function is called when the list of children of the transform of the GameObject has changed.
OnTransformParentChanged	This function is called when the parent property of the transform of the GameObject has changed.
OnTriggerEnter	OnTriggerEnter is called when the Collider other enters the trigger.
OnTriggerEnter2D	Sent when another object enters a trigger collider attached to this object (2D physics only).
OnTriggerExit	OnTriggerExit is called when the Collider other has stopped touching

	the trigger.
OnTriggerExit2D	Sent when another object leaves a trigger collider attached to this object (2D physics only).
OnTriggerStay	OnTriggerStay is called once per physics update for every Collider other that is touching the trigger.
OnTriggerStay2D	Sent each frame where another object is within a trigger collider attached to this object (2D physics only).
OnValidate	This function is called when the script is loaded or a value is changed in the inspector (Called in the editor only).
OnWillRenderObject	OnWillRenderObject is called for each camera if the object is visible and not a UI element.
Reset	Reset to default values.
Start	Start is called on the frame when a script is enabled just before any of the Update methods are called the first time.
Update	Update is called every frame, if the MonoBehaviour is enabled.

Description

Structure describing a bounded plane representing a real-world surface.

Unlike a Plane, which is infinite, a BoundedPlane has a size and a more detailed shape. A BoundedPlane can change from frame to frame (see XRPlaneSubsystem.PlaneUpdated). Typically, this means the current understanding of the plane has changed, rather than the real-world geometry. This can happen as the device explores the physical space and is able to perceive more detail.

Properties

Alignment	The alignment of the plane, e.g., horizontal or vertical.
Center	Center point of the plane in device space.
Height	Current height of the plane.
Id	A session-unique identifier for the plane.
Normal	Normal vector of the plane in device space.
Plane	Returns the infinite Plane associated with this BoundedPlane.
Pose	Pose of the plane in device space.
Size	Current size of the plane.
SubsumedById	A session-unique identifier for the BoundedPlane that subsumed this plane.
Width	Current width of the plane.

Public Methods

GetCorners	Outputs four points, in device space, representing the four corners of the plane. The corners are in clockwise order.
TryGetBoundary	Try to retrieve a list of positions in device space describing current plane boundary.

Description

Structure containing data passed during Frame Received Event.

Properties

CameraSubsystem	Reference to the XRCameraSubsystem associated with this event.

Description

Contains data supplied to a XRPlaneSubsystem.PlaneAdded event.

See Also: XRPlaneSubsystem, XRPlaneSubsystem.PlaneAdded

Properties

Plane	The BoundedPlane that was added.
PlaneSubsystem	A reference to the PlaneSubsystem object from which the event originated.

Description

Contains data supplied to a XRPlaneSubsystem.PlaneRemoved event.

See Also: XRPlaneSubsystem, XRPlaneSubsystem.PlaneRemoved

Properties

Plane	The BoundedPlane that was removed.
PlaneSubsystem	A reference to the XRPlaneSubsystem object from which the event originated.

Description

Contains data supplied to a XRPlaneSubsystem.PlaneUpdated event.

See Also: XRPlaneSubsystem, XRPlaneSubsystem.PlaneUpdated

Properties

Plane	The BoundedPlane that was updated.
PlaneSubsystem	A reference to the XRPlaneSubsystem object from which the event originated.

Description

Contains data supplied to a XRDepth.PointCloudUpdated event.

See Also: XRDepth, XRDepth.PointCloudUpdated

Properties

DepthSubsystem	A reference to the XRDepthSubsystem object from which the event originated.

Description

Describes the transform data of a real-world point.

To track and stabilize around important points in the real world, add a ReferencePoint at the point of interest, and respond to updates to that ReferencePoint. Each frame, each ReferencePoint may have its position and rotation updated (see Environment.ReferencePointUpdated) as the device gains a more thorough understanding of where that point is in Unity space.

Properties

Id	ID for the reference point that is unique across the session.
Pose	The pose (position and rotation) of the reference point. Respond to changes in this pose to correct for changes in the device's understanding of where this point is in the real world.
TrackingState	The TrackingState of the reference point.

Description

Data to be passed to the user when the device corrects its understanding of the world enough that the ReferencePoint's position or rotation has changed.

Properties

PreviousPose	The previous Pose of the ReferencePoint, prior to this event.
PreviousTrackingState	The previous TrackingState of the ReferencePoint, prior to this event.
ReferencePoint	The reference point that has the value of its position, rotation, or both changed enough through the device correcting its understanding of where this point should be located in device space.

Description

Structure defining Tracking State Changed event arguments passed when tracking state changes.

Properties

NewState	New Tracking State.
SessionSubsystem	Reference to the XRSessionSubsystem object associated with the event.

Description

A session-unique identifier for trackables in the environment, e.g., planes and feature points.

Ids are generally unique to a particular XRSessionSubsystem, but multiple sessions may produce identical ids for different trackables.

Static Properties

InvalidId	Represents an invalid id.

Public Methods

ToString	Generates a nicely formatted version of the id.

Description

Provides access to a device's camera.

The XRCameraSubsystem links a Unity Camera to a device camera for video overlay (pass-thru rendering). It also allows developers to query for environmental light estimation, when available.

Properties

Camera	Set current Camera component within the app to be used by this XRCameraInstance.
LastUpdatedFrame	The frame during which the camera subsystem was last successfully updated.
LightEstimationRequested	True if the XRCameraSubsystem should try to provide light estimation.
Material	Set current Material to be used while rendering to the render target.

Public Methods

GetTextures	Fills the provided texturesOut with the texture(s) associated with the XRCameraSubsystem.
TryGetAverageBrightness	Provides brightness for the whole image as an average of all pixels' brightness.
TryGetAverageColorTemperature	Provides color temperature for the whole image as an average of all pixels' color temperature.
TryGetDisplayMatrix	Provides display matrix defining how texture is being rendered on the screen.
TryGetProjectionMatrix	Provides projection matrix used by camera subsystem.
TryGetShaderName	Provides shader name used by Camera subsystem to render texture.
TryGetTimestamp	Provides timestamp.

Events

FrameReceived	Event raised when a new camera frame is received.

Inherited Members

Public Methods

Destroy	Destroys this instance of a subsystem.
Start	Starts an instance of a subsystem.
Stop	Stops an instance of a subsystem.

Description

Class providing information about XRCameraSubsystem registration.

Properties

ProvidesAverageBrightness	Specifies if current subsystem is allowed to provide average brightness.
ProvidesAverageColorTemperature	Specifies if current subsystem is allowed to provide average camera temperature.
ProvidesDisplayMatrix	Specifies if current subsystem is allowed to provide display matrix.
ProvidesProjectionMatrix	Specifies if current subsystem is allowed to provide projection matrix.
ProvidesTimestamp	Specifies if current subsystem is allowed to provide timestamp.

Inherited Members

Properties

id	A unique string that identifies the subsystem that this Descriptor can create.

Description

Provides access to depth data of the physical environment, such as a point cloud.

Properties

LastUpdatedFrame	The frame during which the point cloud was last updated.

Public Methods

GetConfidence	Retrieves the confidence values for each point in the point cloud.
GetPoints	Retrieves the point cloud points.

Events

PointCloudUpdated	Raised once during each frame in which the point cloud is updated.

Inherited Members

Public Methods

Destroy	Destroys this instance of a subsystem.
Start	Starts an instance of a subsystem.
Stop	Stops an instance of a subsystem.

Description

Class providing information about XRDepthSubsystem registration.

Properties

SupportsFeaturePoints	When true, XRDepthSubsystem will provide list of feature points detected so far.

Inherited Members

Properties

id	A unique string that identifies the subsystem that this Descriptor can create.

Description

XRInputSubsystem Instance is used to enable and disable the inputs coming from a specific plugin.

Starting up an XRInputSubsystem Instance will cause that plugin to start feeding input device data to the following InputTracking systems: InputTracking.GetLocalPosition and InputTracking.GetLocalRotation. Calling the Stop or Shutdown functions will disable polling any input device data for that plugin.

Inherited Members

Public Methods

Destroy	Destroys this instance of a subsystem.
Start	Starts an instance of a subsystem.
Stop	Stops an instance of a subsystem.

Description

Information about an Input subsystem.

Properties

disablesLegacyInput	When true, will suppress legacy support for Daydream, Oculus, OpenVR, and Windows MR built directly into the Unity runtime from generating input. This is useful when adding an XRInputSubsystem that supports these devices.

Inherited Members

Properties

id	A unique string that identifies the subsystem that this Descriptor can create.

Description

Provides methods, events, and properties that provides information about planes detected in the environment.

Properties

LastUpdatedFrame	The frame during which the planes were last updated.

Public Methods

GetAllPlanes	Get all the BoundedPlanes currently tracked by the system.
TryGetPlane	Get a BoundedPlane by TrackableId
TryGetPlaneBoundary	Try to retrieve a list of positions in device space describing the current plane boundary.

Events

PlaneAdded	Raised for each BoundedPlane that has been added in the current frame.
PlaneRemoved	Raised for each BoundedPlane that has been removed in the current frame.
PlaneUpdated	Raised for each plane that has been updated in the current frame.

Inherited Members

Public Methods

Destroy	Destroys this instance of a subsystem.
Start	Starts an instance of a subsystem.
Stop	Stops an instance of a subsystem.

Description

Class providing information about XRPlaneSubsystem registration.

Inherited Members

Properties

id	A unique string that identifies the subsystem that this Descriptor can create.

Description

Structure describing the result of a XRRaycastSubsystem.Raycast hit.

See Also: XRRaycastSubsystem, BoundedPlane, TrackableType, TrackableId

Properties

Distance	The distance, in meters, from the screen to the hit's XRRaycastSubsystemHit.Position.
HitType	The TrackableType(s) that were hit.
Pose	The position and rotation of the hit result in device space where the ray hit the trackable.
TrackableId	The TrackableId of the trackable that was hit by the raycast.

Description

Provides methods and properties that allow for querying portions of the physical environment that are near a provided specified ray. These trackables include planes and depth data.

Public Methods

Raycast	Casts a ray from a screen point against selected trackables (e.g., planes and feature points).

Inherited Members

Public Methods

Destroy	Destroys this instance of a subsystem.
Start	Starts an instance of a subsystem.
Stop	Stops an instance of a subsystem.

Description

Class providing information about XRRaycastSubsystem registration.

Inherited Members

Properties

id	A unique string that identifies the subsystem that this Descriptor can create.

Description

Provides methods and properties that allow for querying, creating, and removing of reference points. These reference points are cues to the XRSessionSubsystem that indicate areas of interest in the environment which helps assure that tracking of these points remains accurate.

Properties

LastUpdatedFrame	The frame during which the reference points were last updated.

Public Methods

GetAllReferencePoints	Retrieves all ReferencePoints added by calls to XRReferencePointSubsystem.TryAddReferencePoint.
TryAddReferencePoint	Attempt to add a ReferencePoint that gets tracked by the device.
TryGetReferencePoint	Attempt to retrieve a ReferencePoint.
TryRemoveReferencePoint	Attempt to remove a ReferencePoint getting tracked by the device.

Events

ReferencePointUpdated	Raised each frame for each ReferencePoint that had the values of its position, rotation, or both changed enough by the device correcting its understanding of where the point should be located in Unity space.

Inherited Members

Public Methods

Destroy	Destroys this instance of a subsystem.
Start	Starts an instance of a subsystem.
Stop	Stops an instance of a subsystem.

Description

Class providing information about XRReferencePointSubsystem registration.

Inherited Members

Properties

id	A unique string that identifies the subsystem that this Descriptor can create.

Description

A collection of methods and properties used to interact with and configure an XR session.

Properties

LastUpdatedFrame	The frame during which the tracking state was last updated.
TrackingState	Get current tracking status of the device.

Events

TrackingStateChanged	Raised when the TrackingState changes.

Inherited Members

Public Methods

Destroy	Destroys this instance of a subsystem.
Start	Starts an instance of a subsystem.
Stop	Stops an instance of a subsystem.

Description

Class providing information about XRSessionSubsystem registration.

Inherited Members

Properties

id	A unique string that identifies the subsystem that this Descriptor can create.

Description

Describes current plane alignment in mixed reality space.

Properties

Horizontal	Plane has horizontal alignment.
Vertical	Plane has vertical alignment.
NonAxis	Plane is not alligned along cardinal (X, Y or Z) axis.

Description

A trackable is feature in the physical environment that a device is able to track, such as a plane.

`TrackableType` may be used as a flag and passed to methods like XRRaycastSubsystem.Raycast to specify different types of trackables to raycast against.

Properties

None	No trackable.
PlaneWithinPolygon	The boundary of a BoundedPlane
PlaneWithinBounds	Within the BoundedPlane.Size of a BoundedPlane
PlaneWithinInfinity	The infinite plane of a BoundedPlane
PlaneEstimated	An estimated plane.
Planes	Any of the plane types.
FeaturePoint	A feature point.
All	All trackables (planes and point cloud)

Description

Current tracking state of the device.

Tracking refers to the ability of the device to know its position and orientation in space. Covering the device's sensors, for example, may cause it to lose tracking.

Properties

Unknown	Tracking state is unknown.
Tracking	Tracking is currently working.
Unavailable	Tracking is not available.

Implements interfaces:ISubsystem

Description

A Subsystem is initialized from a SubsystemDescriptor for a given Subsystem (Example, Input, Environment, Display, etc.) and provides an interface to interact with that given Subsystem until it is Destroyed. After a Subsystem is created it can be Started or Stopped to turn on and off functionality (and preserve performance). The base type for Subsystem only exposes this functionality; this class is designed to be a base class for derived classes that expose more functionality specific to a given Subsystem.

Note: initializing a second Subsystem from the same SubsystemDescriptor will return a reference to the existing Subsystem as only one Subsystem is currently allowed for a single Subsystem provider.

Public Methods

Destroy	Destroys this instance of a subsystem.
Start	Starts an instance of a subsystem.
Stop	Stops an instance of a subsystem.

Description

Gives access to subsystems which provide additional functionality through plugins.

Provides the ability to query for SubsystemDescriptors which enumerate features. Given an SubsystemDescriptor, you can create an Subsystem to utilize the subsystem.

Static Methods

GetInstances	Returns active Subsystems of a specific instance type.
GetSubsystemDescriptors	Returns a list of SubsystemDescriptors which describe additional functionality that can be enabled.

Obsolete

Description

ADBannerView is a wrapper around the ADBannerView class found in the Apple iAd framework and is only available on iOS.

It provides a view that displays banner advertisements to the user.

iOS.ADBannerView class is obsolete. Apple iAD service discontinued.

Obsolete

Description

Specifies how banner should be layed out on screen.

Please note that Traditional Banner and Rect Banner (introduced in ios6) use different enum values. If you use "wrong" enum value for current ad type: for Traditional Banner Top* and Center* will be considered Top, and Bottom* will be considered Bottom* for Rect Banner Top will be considered TopLeft and Bottom - BottomLeft.

Obsolete

Description

The type of the banner view.

Please check availability of banner type with ADBannerView.IsAvailable. Traditional banner is always available, though.

Obsolete

Description

ADInterstitialAd is a wrapper around the ADInterstitialAd class found in the Apple iAd framework and is only available on iPad.

It provides full-screen advertisements that your iPad application can display to the user.

Please note that this class is a thin wrapper around the iOS native iAD class, so care should be taken when creating/destroying instances of it frequently. If you need to show ads frequently, instead of constantly destroying and recreating the object you should manually call ReloadAd, or create an auto-reloading ADInterstitialAd.

iOS.ADInterstitialAd class is obsolete. Apple iAD service discontinued.

Description

Interface into iOS specific functionality.

Static Properties

advertisingIdentifier	Advertising ID.
advertisingTrackingEnabled	Is advertising tracking enabled.
deferSystemGesturesMode	Defer system gestures until the second swipe on specific edges.
generation	The generation of the device. (Read Only)
hideHomeButton	Specifies whether the home button should be hidden in the iOS build of this application.
systemVersion	iOS version.
vendorIdentifier	Vendor ID.

Static Methods

RequestStoreReview	Request App Store rating and review from the user.
ResetNoBackupFlag	Reset "no backup" file flag: file will be synced with iCloud/iTunes backup and can be deleted by OS in low storage situations.
SetNoBackupFlag	Set file flag to be excluded from iCloud/iTunes backup.

Description

iOS.LocalNotification is a wrapper around the UILocalNotification class found in the Apple UIKit framework and is only available on iPhone/iPad/iPod Touch.

It represents notifications that an application can schedule for presentation to its user at specific dates and times. The operating system is responsible for delivering the notification at the specified time. Local notifications are similar to remote notifications but are scheduled and delivered locally and do not require connection with remote servers.

Local notifications are scheduled and handled using NotificationServices class.

Static Properties

defaultSoundName	The default system sound. (Read Only)

Properties

alertAction	The title of the action button or slider.
alertBody	The message displayed in the notification alert.
alertLaunchImage	Identifies the image used as the launch image when the user taps the action button.
alertTitle	A short description of the reason for the alert.
applicationIconBadgeNumber	The number to display as the application's icon badge.
fireDate	The date and time when the system should deliver the notification.
hasAction	A boolean value that controls whether the alert action is visible or not.
repeatCalendar	The calendar type (Gregorian, Chinese, etc) to use for rescheduling the notification.
repeatInterval	The calendar interval at which to reschedule the notification.
soundName	The name of the sound file to play when an alert is displayed.
timeZone	The time zone of the notification's fire date.
userInfo	A dictionary for passing custom information to the notified application.

Constructors

LocalNotification	Creates a new local notification.

Description

NotificationServices is only available on iPhone/iPad/iPod Touch.

It is used for scheduling, registering and handling notifications.

Static Properties

deviceToken	Device token received from Apple Push Service after calling NotificationServices.RegisterForRemoteNotificationTypes. (Read Only)
enabledNotificationTypes	Enabled local and remote notification types.
localNotificationCount	The number of received local notifications. (Read Only)
localNotifications	The list of objects representing received local notifications. (Read Only)
registrationError	Returns an error that might occur on registration for remote notifications via NotificationServices.RegisterForRemoteNotificationTypes. (Read Only)
remoteNotificationCount	The number of received remote notifications. (Read Only)
remoteNotifications	The list of objects representing received remote notifications. (Read Only)
scheduledLocalNotifications	All currently scheduled local notifications.

Static Methods

CancelAllLocalNotifications	Cancels the delivery of all scheduled local notifications.
CancelLocalNotification	Cancels the delivery of the specified scheduled local notification.
ClearLocalNotifications	Discards of all received local notifications.
ClearRemoteNotifications	Discards of all received remote notifications.
GetLocalNotification	Returns an object representing a specific local notification. (Read Only)
GetRemoteNotification	Returns an object representing a specific remote notification. (Read Only)
PresentLocalNotificationNow	Presents a local notification immediately.
RegisterForNotifications	Register to receive local and remote notifications of the specified types from a provider via Apple Push Service.
ScheduleLocalNotification	Schedules a local notification.
UnregisterForRemoteNotifications	Unregister for remote notifications.

Description

On Demand Resources API.

See Also: OnDemandResources.PreloadAsync and OnDemandResourcesRequest.

Static Properties

enabled	Indicates whether player was built with "Use On Demand Resources" player setting enabled.

Static Methods

PreloadAsync	Creates an On Demand Resources (ODR) request.

Description

Represents a request for On Demand Resources (ODR). It's an AsyncOperation and can be yielded in a coroutine.

NOTE: only available on iOS.

Creating an OnDemandResourcesRequest is equivalent to calling NSBundleResourceRequest.beginAccessingResourcesWithCompletionHandler . The request will keep the on demand resource alive until either Dispose() is called or the request object is collected by a garbage collector, which is the equivalent of calling NSBundleResourceRequest.endAccessingResources .

```
using UnityEngine;
using UnityEngine.iOS;
using System;
using System.Collections;


public static class Loader
{
    public static IEnumerator LoadAsset(string resourceName)
    {
        // Create the request
        var request = OnDemandResources.PreloadAsync(new string[] { "Asset's ODR


        // Wait until request is completed
        yield return request;


        // Check for errors
        if (request.error != null)
            throw new Exception("ODR request failed: " + request.error);


        // Get path to the resource and use it
        var path = request.GetResourcePath(resourceName);
        Debug.Log(path);


        // Call Dispose() when resource is no longer needed.
        request.Dispose();
    }
}
```

Properties

error	Returns an error after operation is complete.
loadingPriority	Sets the priority for request.

Public Methods

Dispose	Release all resources kept alive by On Demand Resources (ODR) request.
GetResourcePath	Gets file system's path to the resource available in On Demand Resources (ODR) request.

Inherited Members

Properties

allowSceneActivation	Allow Scenes to be activated as soon as it is ready.
isDone	Has the operation finished? (Read Only)
priority	Priority lets you tweak in which order async operation calls will be performed.
progress	What's the operation's progress. (Read Only)

Events

completed	Event that is invoked upon operation completion. An event handler that is registered in the same frame as the call that creates it will be invoked next frame, even if the operation is able to complete synchronously. If a handler is registered after the operation has completed and has already invoked the complete event, the handler will be called synchronously.

Description

RemoteNotification is only available on iPhone/iPad/iPod Touch.

Remote notifications are sent by an application's remote server (its provider) to Apple Push Notification service which pushes the notification to devices on which the application is installed.

Remote notifications are registered for and handled using NotificationServices class.

Properties

alertBody	The message displayed in the notification alert. (Read Only)
alertTitle	A short description of the reason for the alert. (Read Only)
applicationIconBadgeNumber	The number to display as the application's icon badge. (Read Only)
hasAction	A boolean value that controls whether the alert action is visible or not. (Read Only)
soundName	The name of the sound file to play when an alert is displayed. (Read Only)
userInfo	A dictionary for passing custom information to the notified application. (Read Only)

Description

ActivityIndicator Style (iOS Specific).

Properties

DontShow	Do not show ActivityIndicator.
WhiteLarge	The large white style of indicator (UIActivityIndicatorViewStyleWhiteLarge).
White	The standard white style of indicator (UIActivityIndicatorViewStyleWhite).
Gray	The standard gray style of indicator (UIActivityIndicatorViewStyleGray).

Description

Specify calendar types.

Properties

GregorianCalendar	Identifies the Gregorian calendar.
BuddhistCalendar	Identifies the Buddhist calendar.
ChineseCalendar	Identifies the Chinese calendar.
HebrewCalendar	Identifies the Hebrew calendar.
IslamicCalendar	Identifies the Islamic calendar.
IslamicCivilCalendar	Identifies the Islamic civil calendar.
JapaneseCalendar	Identifies the Japanese calendar.
RepublicOfChinaCalendar	Identifies the Republic of China (Taiwan) calendar.
PersianCalendar	Identifies the Persian calendar.
IndianCalendar	Identifies the Indian calendar.
ISO8601Calendar	Identifies the ISO8601.

Description

Specify calendrical units.

Properties

Era	Specifies the era unit.
Year	Specifies the year unit.
Month	Specifies the month unit.
Day	Specifies the day unit.
Hour	Specifies the hour unit.
Minute	Specifies the minute unit.
Second	Specifies the second unit.
Week	Specifies the week unit.
Weekday	Specifies the weekday unit.
WeekdayOrdinal	Specifies the ordinal weekday unit.
Quarter	Specifies the quarter of the calendar.

Description

iOS device generation.

Properties

iPhone	iPhone, first generation.
iPhone3G	iPhone, second generation.
iPhone3GS	iPhone, third generation.
iPodTouch1Gen	iPod Touch, first generation.
iPodTouch2Gen	iPod Touch, second generation.
iPodTouch3Gen	iPod Touch, third generation.
iPad1Gen	iPad, first generation.
iPhone4	iPhone, fourth generation.
iPodTouch4Gen	iPod Touch, fourth generation.
iPad2Gen	iPad, second generation.
iPhone4S	iPhone, fifth generation.
iPad3Gen	iPad, third generation.
iPhone5	iPhone5.
iPodTouch5Gen	iPod Touch, fifth generation.
iPadMini1Gen	iPadMini, first generation.
iPad4Gen	iPad, fourth generation.
iPhone5C	iPhone 5C.
iPhone5S	iPhone 5S.
iPadAir1	iPad Air.
iPadMini2Gen	iPadMini Retina, second generation.
iPhone6	iPhone 6.
iPhone6Plus	iPhone 6 plus.
iPadMini3Gen	iPad Mini 3.

iPadAir2	iPad Air 2.
iPhone6S	iPhone 6S.
iPhone6SPlus	iPhone 6S Plus.
iPadPro1Gen	iPad Pro 12.9", first generation.
iPadMini4Gen	iPad Mini, fourth generation.
iPhoneSE1Gen	iPhone SE, first generation.
iPadPro10Inch1Gen	iPad Pro 9.7", first generation.
iPhone7	iPhone 7.
iPhone7Plus	iPhone 7 Plus.
iPodTouch6Gen	iPod Touch, sixth generation.
iPad5Gen	iPad Air, fifth generation.
iPadPro2Gen	iPad Pro 12.9", second generation.
iPadPro10Inch2Gen	iPad Pro 10.5", second generation 10" iPad.
iPhone8	iPhone 8.
iPhone8Plus	iPhone 8 Plus.
iPhoneX	iPhone X.
iPhoneXS	iPhone XS.
iPhoneXSMax	iPhone XSMax.
iPhoneXR	iPhone XR.
iPhoneUnknown	Yet unknown iPhone.
iPadUnknown	Yet unknown iPad.
iPodTouchUnknown	Yet unknown iPod Touch.

Description

Specifies local and remote notification types.

Properties

None	No notification types specified.
Badge	Notification is a badge shown above the application's icon.
Sound	Notification is an alert sound.
Alert	Notification is an alert message.

Description

Bit-mask used to control the deferring of system gestures on iOS.

On iPhone X the home button is implemented as a system gesture (swipe up from the lower edge of the screen). Other gestures are implemented as swipes from other screen edges. Note, this may interfere with games that use swipes as an interaction method. This bit-mask specifies which screen edge gestures the system defers to the second swipe. If deferring is enabled on a specific screen edge, the first swipe is ignored and provides a way to reduce unwanted interactions with the App. Note that iOS Human interface guidelines do not recommend enabling this behavior as it may confuse users.

Properties

None	Disables gesture deferring on all edges.
TopEdge	Identifies top screen edge.
LeftEdge	Identifies left screen edge.
BottomEdge	Identifies bottom screen edge.
RightEdge	Identifies right screen edge.
All	Identifies all screen edges.

Description

Extension methods for IJobParallelForTransform.

Static Methods

Schedule	Schedule.

Description

Position, rotation and scale of an object.

Properties

localPosition	The position of the transform relative to the parent.
localRotation	The rotation of the transform relative to the parent transform's rotation.
localScale	The scale of the transform relative to the parent.
position	The position of the transform in world space.
rotation	The rotation of the transform in world space stored as a Quaternion.

Description

TransformAccessArray.

Properties

capacity	Returns array capacity.
isCreated	isCreated.
length	Length.
this[int]	Array indexer.

Constructors

TransformAccessArray	Constructor.

Public Methods

Add	Add.
Dispose	Dispose.
RemoveAtSwapBack	Remove item at index.
SetTransforms	Set transforms.

Static Methods

Allocate	Allocate.

Description

IJobParallelForTransform.

Public Methods

Execute	Execute.

Obsolete

Description

Details about a UNET MatchMaker match.

This is a data object that represents the data returned from NetworkMatch.CreateMatch and NetworkMatch.JoinMatch.

Obsolete

Description

A class describing the match information as a snapshot at the time the request was processed on the MatchMaker.

The information in this class is part of the data list returned when listing open matches through MatchMaker. It describes the instantaneous state at the time the request was handled.

Obsolete

Description

A class describing one member of a match and what direct connect information other clients have supplied.

A "MatchInfoDirectConnectSnapshot" instance contains mostly optional data which a connection can supply when it creates or joins a match. That info is then returned in these results so that other clients can directly connect to that client instead of going through the internet via a relay server.

Obsolete

Description

A component for communicating with the Unity Multiplayer Matchmaking service.

This class aggregates all the supported JSON calls into functions that are callable for games that wish to interact with the MatchMaker service.

The example code below shows how to use NetworkMatch API.

```
using UnityEngine;
using UnityEngine.Networking;
using UnityEngine.Networking.Match;
using System.Collections.Generic;


public class SimpleMatchMaker : MonoBehaviour
{
    void Start()
    {
        NetworkManager.singleton.StartMatchMaker();
    }


    //call this method to request a match to be created on the server
    public void CreateInternetMatch(string matchName)
    {
        NetworkManager.singleton.matchMaker.CreateMatch(matchName, 4, true, "",
    }


    //this method is called when your request for creating a match is returned
    private void OnInternetMatchCreate(bool success, string extendedInfo, MatchI
    {
        if (success)
        {
            //Debug.Log("Create match succeeded");


            MatchInfo hostInfo = matchInfo;
            NetworkServer.Listen(hostInfo, 9000);
```

```
        NetworkManager.singleton.StartHost(hostInfo);
    }
    else
    {
        Debug.LogError("Create match failed");
    }
}


//call this method to find a match through the matchmaker
public void FindInternetMatch(string matchName)
{
    NetworkManager.singleton.matchMaker.ListMatches(0, 10, matchName, true,
}


//this method is called when a list of matches is returned
private void OnInternetMatchList(bool success, string extendedInfo, List<Mat
{
    if (success)
    {
        if (matches.Count != 0)
        {
            //Debug.Log("A list of matches was returned");


            //join the last server (just in case there are two...)
            NetworkManager.singleton.matchMaker.JoinMatch(matches[matches.Co
        }
        else
        {
            Debug.Log("No matches in requested room!");
        }
    }
    else
    {
        Debug.LogError("Couldn't connect to match maker");
    }
}


//this method is called when your request to join a match is returned
private void OnJoinInternetMatch(bool success, string extendedInfo, MatchInf
{
    if (success)
    {
```

```
            //Debug.Log("Able to join a match");


            MatchInfo hostInfo = matchInfo;
            NetworkManager.singleton.StartClient(hostInfo);
        }
        else
        {
            Debug.LogError("Join match failed");
        }
    }
}
```

Inherited Members

Properties

enabled	Enabled Behaviours are Updated, disabled Behaviours are not.
isActiveAndEnabled	Has the Behaviour had active and enabled called?
gameObject	The game object this component is attached to. A component is always attached to a game object.
tag	The tag of this game object.
transform	The Transform attached to this GameObject.
runInEditMode	Allow a specific instance of a MonoBehaviour to run in edit mode (only available in the editor).
useGUILayout	Disabling this lets you skip the GUI layout phase.
hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.

Public Methods

BroadcastMessage	Calls the method named methodName on every MonoBehaviour in this game object or any of its children.
CompareTag	Is this game object tagged with tag ?
GetComponent	Returns the component of Type type if the game object has one attached, null if it doesn't.
GetComponentInChildren	Returns the component of Type type in the GameObject or any of its children

	using depth first search.
GetComponentInParent	Returns the component of Type type in the GameObject or any of its parents.
GetComponents	Returns all components of Type type in the GameObject.
GetComponentsInChildren	Returns all components of Type type in the GameObject or any of its children.
GetComponentsInParent	Returns all components of Type type in the GameObject or any of its parents.
SendMessage	Calls the method named methodName on every MonoBehaviour in this game object.
SendMessageUpwards	Calls the method named methodName on every MonoBehaviour in this game object and on every ancestor of the behaviour.
CancelInvoke	Cancels all Invoke calls on this MonoBehaviour.
Invoke	Invokes the method methodName in time seconds.
InvokeRepeating	Invokes the method methodName in time seconds, then repeatedly every repeatRate seconds.
IsInvoking	Is any invoke on methodName pending?
StartCoroutine	Starts a coroutine.
StopAllCoroutines	Stops all coroutines running on this behaviour.
StopCoroutine	Stops the first coroutine named methodName, or the coroutine stored in routine running on this behaviour.
GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.

Static Methods

print	Logs message to the Unity Console (identical to Debug.Log).
Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Messages

Awake	Awake is called when the script instance is being loaded.
FixedUpdate	This function is called every fixed framerate frame, if the MonoBehaviour is enabled.
LateUpdate	LateUpdate is called every frame, if the Behaviour is enabled.
OnAnimatorIK	Callback for setting up animation IK (inverse kinematics).
OnAnimatorMove	Callback for processing animation movements for modifying root motion.
OnApplicationFocus	Sent to all GameObjects when the player gets or loses focus.
OnApplicationPause	Sent to all GameObjects when the application pauses.
OnApplicationQuit	Sent to all game objects before the application quits.
OnAudioFilterRead	If OnAudioFilterRead is implemented, Unity will insert a custom filter into the audio DSP chain.
OnBecameInvisible	OnBecameInvisible is called when the renderer is no longer visible by any camera.
OnBecameVisible	OnBecameVisible is called when the renderer became visible by any camera.
OnCollisionEnter	OnCollisionEnter is called when this collider/rigidbody has begun touching another rigidbody/collider.
OnCollisionEnter2D	Sent when an incoming collider makes contact with this object's collider (2D physics only).
OnCollisionExit	OnCollisionExit is called when this collider/rigidbody has stopped touching another rigidbody/collider.
OnCollisionExit2D	Sent when a collider on another object stops touching this object's collider (2D physics only).
OnCollisionStay	:ref::OnCollisionStay is called once per frame for every collider/rigidbody that is touching rigidbody/collider.

OnCollisionStay2D	Sent each frame where a collider on another object is touching this object's collider (2D physics only).
OnConnectedToServer	Called on the client when you have successfully connected to a server.
OnControllerColliderHit	OnControllerColliderHit is called when the controller hits a collider while performing a Move.
OnDestroy	Destroying the attached Behaviour will result in the game or Scene receiving OnDestroy.
OnDisable	This function is called when the behaviour becomes disabled.
OnDisconnectedFromServer	Called on the client when the connection was lost or you disconnected from the server.
OnDrawGizmos	Implement OnDrawGizmos if you want to draw gizmos that are also pickable and always drawn.
OnDrawGizmosSelected	Implement OnDrawGizmosSelected to draw a gizmo if the object is selected.
OnEnable	This function is called when the object becomes enabled and active.
OnFailedToConnect	Called on the client when a connection attempt fails for some reason.
OnFailedToConnectToMasterServer	Called on clients or servers when there is a problem connecting to the MasterServer.
OnGUI	OnGUI is called for rendering and handling GUI events.
OnJointBreak	Called when a joint attached to the same game object broke.
OnJointBreak2D	Called when a Joint2D attached to the same game object breaks.
OnMasterServerEvent	Called on clients or servers when reporting events from the MasterServer.
OnMouseDown	OnMouseDown is called when the user has pressed the mouse button while over the GUIElement or Collider.
OnMouseDrag	OnMouseDrag is called when the user has clicked on a GUIElement or Collider and is still holding down the mouse.
OnMouseEnter	Called when the mouse enters the GUIElement or Collider.
OnMouseExit	Called when the mouse is not any longer over the GUIElement or Collider.
OnMouseOver	Called every frame while the mouse is over the GUIElement or Collider.
OnMouseUp	OnMouseUp is called when the user has released the mouse button.

OnMouseUpAsButton	OnMouseUpAsButton is only called when the mouse is released over the same GUIElement or Collider as it was pressed.
OnNetworkInstantiate	Called on objects which have been network instantiated with Network.Instantiate.
OnParticleCollision	OnParticleCollision is called when a particle hits a Collider.
OnParticleSystemStopped	OnParticleSystemStopped is called when all particles in the system have died, and no new particles will be born. New particles cease to be created either after Stop is called, or when the duration property of a non-looping system has been exceeded.
OnParticleTrigger	OnParticleTrigger is called when any particles in a particle system meet the conditions in the trigger module.
OnPlayerConnected	Called on the server whenever a new player has successfully connected.
OnPlayerDisconnected	Called on the server whenever a player disconnected from the server.
OnPostRender	OnPostRender is called after a camera finished rendering the Scene.
OnPreCull	OnPreCull is called before a camera culls the Scene.
OnPreRender	OnPreRender is called before a camera starts rendering the Scene.
OnRenderImage	OnRenderImage is called after all rendering is complete to render image.
OnRenderObject	OnRenderObject is called after camera has rendered the Scene.
OnSerializeNetworkView	Used to customize synchronization of variables in a script watched by a network view.
OnServerInitialized	Called on the server whenever a Network.InitializeServer was invoked and has completed.
OnTransformChildrenChanged	This function is called when the list of children of the transform of the GameObject has changed.
OnTransformParentChanged	This function is called when the parent property of the transform of the GameObject has changed.
OnTriggerEnter	OnTriggerEnter is called when the Collider other enters the trigger.
OnTriggerEnter2D	Sent when another object enters a trigger collider attached to this object (2D physics only).
OnTriggerExit	OnTriggerExit is called when the Collider other has stopped touching the trigger.
OnTriggerExit2D	Sent when another object leaves a trigger collider attached to this object (2D physics only).

OnTriggerStay	OnTriggerStay is called once per physics update for every Collider other that is touching the trigger.
OnTriggerStay2D	Sent each frame where another object is within a trigger collider attached to this object (2D physics only).
OnValidate	This function is called when the script is loaded or a value is changed in the inspector (Called in the editor only).
OnWillRenderObject	OnWillRenderObject is called for each camera if the object is visible and not a UI element.
Reset	Reset to default values.
Start	Start is called on the frame when a script is enabled just before any of the Update methods are called the first time.
Update	Update is called every frame, if the MonoBehaviour is enabled.

Obsolete

Description

This is passed to handler funtions registered for the AddPlayer built-in message.

Inherited Members

Obsolete

Description

A utility class to send a network message with no contents.

```
using UnityEngine;
using UnityEngine.Networking;
using UnityEngine.Networking.NetworkSystem;


public class Test
{
    void SendNotification()
    {
        var msg = new EmptyMessage();
        NetworkServer.SendToAll(667, msg);
    }
}
```

Inherited Members

Obsolete

Description

This is passed to handler functions registered for the SYSTEM_ERROR built-in message.

Inherited Members

Obsolete

Description

A utility class to send simple network messages that only contain an integer.

```
using UnityEngine;
using UnityEngine.Networking;
using UnityEngine.Networking.NetworkSystem;


public class Test
{
    void SendValue(int value)
    {
        var msg = new IntegerMessage(value);
        NetworkServer.SendToAll(MsgType.Scene, msg);
    }
}
```

Inherited Members

Obsolete

Description

This is passed to handler funtions registered for the SYSTEM_NOT_READY built-in message.

Inherited Members

Obsolete

Description

Information about a change in authority of a non-player in the same network game.

This information is cached by clients and used during host-migration.

Inherited Members

Obsolete

Description

Information about another participant in the same network game.

This information is cached by clients and used during host-migration.

Inherited Members

Obsolete

Description

A structure used to identify player object on other peers for host migration.

Obsolete

Description

Internal UNET message for sending information about network peers to clients.

Inherited Members

Obsolete

Description

This is passed to handler funtions registered for the SYSTEM_READY built-in message.

Inherited Members

Obsolete

Description

This network message is used when a client reconnect to the new host of a game.

Inherited Members

Obsolete

Description

This is passed to handler funtions registered for the SYSTEM_REMOVE_PLAYER built-in message.

Inherited Members

Obsolete

Description

This is a utility class for simple network messages that contain only a string.

This example sends a message with the name of the Scene.

```
using UnityEngine;
using UnityEngine.Networking;
using UnityEngine.Networking.NetworkSystem;


public class Test
{
    void SendSceneName(string sceneName)
    {
        var msg = new StringMessage(sceneName);
        NetworkServer.SendToAll(MsgType.Scene, msg);
    }
}
```

Inherited Members

Description

Arguments passed to Action callbacks registered in PlayerConnection.

Properties

data	Data that is received.
playerId	The Player ID that the data is received from.

Implements interfaces:IEditorPlayerConnection

Description

Used for handling the network connection from the Player to the Editor.

Sets up events for connecting to and sending data to the Editor.

This can only be used in a class that inherits from MonoBehaviour, Object or ScriptableObject.

Static Properties

instance	Singleton instance.

Properties

isConnected	Returns true when Editor is connected to the player.

Public Methods

BlockUntilRecvMsg	Blocks the calling thread until either a message with the specified messageId is received or the specified time-out elapses.
DisconnectAll	This disconnects all of the active connections.
Register	Registers a listener for a specific message ID, with an Action to be executed whenever that message is received by the Editor. This ID must be the same as for messages sent from EditorConnection.Send().
RegisterConnection	Registers a callback that is invoked when the Editor connects to the Player.
RegisterDisconnection	Registers a callback to be called when Editor disconnects.
Send	Sends data to the Editor.
Unregister	Deregisters a message listener.

Inherited Members

Properties

hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.

Public Methods

GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.

Static Methods

Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.
CreateInstance	Creates an instance of a scriptable object.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Messages

Awake	This function is called when the ScriptableObject script is started.
OnDestroy	This function is called when the scriptable object will be destroyed.
OnDisable	This function is called when the scriptable object goes out of scope.
OnEnable	This function is called when the object is loaded.

Description

Access token used to authenticate a client session for the purposes of allowing or disallowing match operations requested by that client.

Properties

array	Binary field for the actual token.

Public Methods

GetByteString	Accessor to get an encoded string from the m_array data.
IsValid	Checks if the token is a valid set of data with respect to default values (returns true if the values are not default, does not validate the token is a current legitimate token with respect to the server's auth framework).

Description

The AppID identifies the application on the Unity Cloud or UNET servers.

Properties

Invalid	Invalid AppID.

Description

An Enum representing the priority of a client in a match, starting at 0 and increasing.

Properties

Invalid	The Invalid case for a HostPriority. An Invalid host priority is not a valid host.

Description

Describes the access levels granted to this client.

Properties

Invalid	Invalid access level, signifying no access level has been granted/specified.
User	User access level. This means you can do operations which affect yourself only, like disconnect yourself from the match.
Owner	Access level Owner, generally granting access for operations key to the peer host server performing it's work.
Admin	Administration access level, generally describing clearence to perform game altering actions against anyone inside a particular match.

Description

Network ID, used for match making.

Properties

Invalid	Invalid NetworkID.

Description

The NodeID is the ID used in Relay matches to track nodes in a network.

Properties

Invalid	The invalid case of a NodeID.

Description

Identifies a specific game instance.

Properties

Invalid	Invalid SourceID.

Description

Responsible for rejecting or accepting certificates received on https requests.

Note: Custom certificate validation is currently only implemented for the following platforms - Android, iOS, tvOS and desktop platforms.

Public Methods

Dispose	Signals that this [CertificateHandler] is no longer being used, and should clean up any resources it is using.

Protected Methods

ValidateCertificate	Callback, invoked for each leaf certificate sent by the remote server.

Obsolete

Description

Defines parameters of channels.

Obsolete

Description

Class containing constants for default network channels.

Obsolete

Description

A client manager which contains static client information and functions.

This manager contains references to tracked static local objects such as spawner registrations. It also has the default message handlers used by clients when they registered none themselves. The manager handles adding/removing player objects to the game after a client connection has been set as ready.

The ClientScene is a singleton, and it has static convenience methods such as ClientScene.Ready().

The ClientScene is used by the NetworkManager, but it can be used by itself.

As the ClientScene manages player objects on the client, it is where clients request to add players. The NetworkManager does this via the ClientScene automatically when auto-add-players is set, but it can be done through code using the function ClientScene.AddPlayer(). This sends an AddPlayer message to the server and will cause a player object to be created for this client.

Like NetworkServer, the ClientScene understands the concept of the local client. The function ClientScene.ConnectLocalServer() is used to become a host by starting a local client (when a server is already running).

Static Methods

SetLocalObject	NetId is a unique number assigned to all objects with NetworkIdentity components in a game.

Obsolete

Description

This class defines parameters of connection between two peers, this definition includes various timeouts and sizes as well as channel configuration.

The NetworkServer and NetworkClient classes can be configured with instances of this class. This causes the settings in the ConnectionConfig to be applied to the transport layer connection of the NetworkConnections owned by these objects.

The NetworkManager has a ConnectionConfig object that is used to configure servers and clients that it creates.

The default channel connection configuration used by the high level components of the networking system is: * channel 0 - Reliable Sequenced channel * channel 1 - Unreliable channel.

Some of the attributes of this ConnectionConfig instance can be modified using the Inspector UI for the NetworkManager in the Editor.

Important: Some parameters of ConnectionConfig should be exactly the same for connected peers. If they are different, a CRCMismatch error is received when trying to establish a connection.

These parameters are:
- PacketSize
- FragmentSize
- ResendTimeout
- DisconnectTimeout
- ConnectTimeout
- PingTimeout
- ReducedPingTimeout
- AllCostTimeout
- IsAckLong
- UsePlatformSpecificProtocols.

Obsolete

Description

Create configuration for network simulator; You can use this class in editor and developer build only.

Description

Manage and process HTTP response body data received from a remote server.

DownloadHandler objects are helper objects. When attached to a UnityWebRequest, they define how to handle HTTP response body data received from a remote server. Generally, they are used to buffer, stream and/or process response bodies.

DownloadHandler is a base class. Depending on usage scenario, different specialized classes are available. DownloadHandlerBuffer provides basic buffering, while DownloadHandlerTexture and DownloadHandlerAssetBundle provide more efficient solutions for Texture and AssetBundle downloads.

For custom use cases, see DownloadHandlerScript.

See Also: UnityWebRequest, DownloadHandlerBuffer, DownloadHandlerTexture, DownloadHandlerAudioClip, DownloadHandlerAssetBundle, DownloadHandlerScript.

Properties

data	Returns the raw bytes downloaded from the remote server, or null. (Read Only)
isDone	Returns true if this DownloadHandler has been informed by its parent UnityWebRequest that all data has been received, and this DownloadHandler has completed any necessary post-download processing. (Read Only)
text	Convenience property. Returns the bytes from data interpreted as a UTF8 string. (Read Only)

Public Methods

Dispose	Signals that this DownloadHandler is no longer being used, and should clean up any resources it is using.

Protected Methods

CompleteContent	Callback, invoked when all data has been received from the remote server.
GetData	Callback, invoked when the data property is accessed.
GetProgress	Callback, invoked when UnityWebRequest.downloadProgress is accessed.
GetText	Callback, invoked when the text property is accessed.
ReceiveContentLength	Callback, invoked with a Content-Length header is received.

ReceiveData	Callback, invoked as data is received from the remote server.

Description

A DownloadHandler subclass specialized for downloading AssetBundles.

This subclass streams downloaded data into Unity's asset bundle decompression and decoding system on worker threads, providing efficient downloading and processing for AssetBundle objects.

Properties

assetBundle	Returns the downloaded AssetBundle, or null. (Read Only)

Constructors

DownloadHandlerAssetBundle	Standard constructor for non-cached asset bundles.

Protected Methods

GetData	Not implemented. Throws NotSupportedException.
GetText	Not implemented. Throws NotSupportedException.

Static Methods

GetContent	Returns the downloaded AssetBundle, or null.

Inherited Members

Properties

data	Returns the raw bytes downloaded from the remote server, or null. (Read Only)
isDone	Returns true if this DownloadHandler has been informed by its parent UnityWebRequest that all data has been received, and this DownloadHandler has completed any necessary post-download processing. (Read Only)
text	Convenience property. Returns the bytes from data interpreted as a UTF8 string. (Read Only)

Public Methods

Dispose	Signals that this DownloadHandler is no longer being used, and should clean up any

resources it is using.

Protected Methods

CompleteContent	Callback, invoked when all data has been received from the remote server.
GetProgress	Callback, invoked when UnityWebRequest.downloadProgress is accessed.
ReceiveContentLength	Callback, invoked with a Content-Length header is received.
ReceiveData	Callback, invoked as data is received from the remote server.

Description

A DownloadHandler subclass specialized for downloading audio data for use as AudioClip objects.

DownloadHandlerAudioClip stores received data in a pre-allocated Unity AudioClip object. It is optimized for downloading audio data from Web servers, and may perform audio data decompression and decoding on a worker thread.

For use cases where you wish to download an audio clip via HTTP and use it as an AudioClip within Unity, usage of this class is strongly recommended.

Properties

audioClip	Returns the downloaded AudioClip, or null. (Read Only)
compressed	Create AudioClip that is compressed in memory.
streamAudio	Create streaming AudioClip.

Constructors

DownloadHandlerAudioClip	Constructor, specifies what kind of audio data is going to be downloaded.

Protected Methods

GetData	Called by DownloadHandler.data. Returns a copy of the downloaded clip data as raw bytes.

Static Methods

GetContent	Returns the downloaded AudioClip, or null.

Inherited Members

Properties

data	Returns the raw bytes downloaded from the remote server, or null. (Read Only)
isDone	Returns true if this DownloadHandler has been informed by its parent UnityWebRequest that all data has been received, and this DownloadHandler has completed any necessary post-download processing. (Read Only)

text	Convenience property. Returns the bytes from data interpreted as a UTF8 string. (Read Only)

Public Methods

Dispose	Signals that this DownloadHandler is no longer being used, and should clean up any resources it is using.

Protected Methods

CompleteContent	Callback, invoked when all data has been received from the remote server.
GetProgress	Callback, invoked when UnityWebRequest.downloadProgress is accessed.
GetText	Callback, invoked when the text property is accessed.
ReceiveContentLength	Callback, invoked with a Content-Length header is received.
ReceiveData	Callback, invoked as data is received from the remote server.

Description

A general-purpose DownloadHandler implementation which stores received data in a native byte buffer.

This is a general-purpose DownloadHandler subclass. It stores received data in native memory. It will preallocated a data buffer based on any received `Content-Length` header, but will expand its buffer should the actual download size exceed the value of `Content-Length` header (or if a `Content-Length` header is not received).

Note: When accessing DownloadHandler.data or DownloadHandler.text on this subclass, a new byte array or string will be allocated each time the property is accessed.

Constructors

DownloadHandlerBuffer	Default constructor.

Protected Methods

GetData	Returns a copy of the contents of the native-memory data buffer as a byte array.

Static Methods

GetContent	Returns a copy of the native-memory buffer interpreted as a UTF8 string.

Inherited Members

Properties

data	Returns the raw bytes downloaded from the remote server, or null. (Read Only)
isDone	Returns true if this DownloadHandler has been informed by its parent UnityWebRequest that all data has been received, and this DownloadHandler has completed any necessary post-download processing. (Read Only)
text	Convenience property. Returns the bytes from data interpreted as a UTF8 string. (Read Only)

Public Methods

Dispose	Signals that this DownloadHandler is no longer being used, and should clean up any resources it is using.

Protected Methods

CompleteContent	Callback, invoked when all data has been received from the remote server.
GetProgress	Callback, invoked when UnityWebRequest.downloadProgress is accessed.
GetText	Callback, invoked when the text property is accessed.
ReceiveContentLength	Callback, invoked with a Content-Length header is received.
ReceiveData	Callback, invoked as data is received from the remote server.

Description

Download handler for saving the downloaded data to file.

This specialized download handler writes all downloaded bytes directly to file. This can help avoid high memory usage. Note that you cannot get any data out of this download handler; you are expected to work with resulting file after the download is finished.

Properties

removeFileOnAbort	Should the created file be removed if download is aborted (manually or due to an error). Default: false.

Constructors

DownloadHandlerFile	Creates a new instance and a file on disk where downloaded data will be written to.

Inherited Members

Properties

data	Returns the raw bytes downloaded from the remote server, or null. (Read Only)
isDone	Returns true if this DownloadHandler has been informed by its parent UnityWebRequest that all data has been received, and this DownloadHandler has completed any necessary post-download processing. (Read Only)
text	Convenience property. Returns the bytes from data interpreted as a UTF8 string. (Read Only)

Public Methods

Dispose	Signals that this DownloadHandler is no longer being used, and should clean up any resources it is using.

Protected Methods

CompleteContent	Callback, invoked when all data has been received from the remote server.
GetData	Callback, invoked when the data property is accessed.
GetProgress	Callback, invoked when UnityWebRequest.downloadProgress is accessed.

GetText	Callback, invoked when the text property is accessed.
ReceiveContentLength	Callback, invoked with a Content-Length header is received.
ReceiveData	Callback, invoked as data is received from the remote server.

Obsolete

Description

MovieTexture has been deprecated. Refer to the new movie playback solution VideoPlayer.

A specialized DownloadHandler for creating MovieTexture out of downloaded bytes. This class is only available on platforms that support MovieTexture.

Inherited Members

Properties

data	Returns the raw bytes downloaded from the remote server, or null. (Read Only)
isDone	Returns true if this DownloadHandler has been informed by its parent UnityWebRequest that all data has been received, and this DownloadHandler has completed any necessary post-download processing. (Read Only)
text	Convenience property. Returns the bytes from data interpreted as a UTF8 string. (Read Only)

Public Methods

Dispose	Signals that this DownloadHandler is no longer being used, and should clean up any resources it is using.

Protected Methods

CompleteContent	Callback, invoked when all data has been received from the remote server.
GetData	Callback, invoked when the data property is accessed.
GetProgress	Callback, invoked when UnityWebRequest.downloadProgress is accessed.
GetText	Callback, invoked when the text property is accessed.
ReceiveContentLength	Callback, invoked with a Content-Length header is received.
ReceiveData	Callback, invoked as data is received from the remote server.

Description

An abstract base class for user-created scripting-driven DownloadHandler implementations.

On its own, a DownloadHandlerScript object does no useful work.

Instead, it forwards data received from the network to the DownloadHandler.ReceiveData callback method, as well as invoking other useful callbacks for download-related events. By default, a DownloadHandlerScript does nothing; however, you can derive your own subclass from DownloadHandlerScript, override some or all of its callbacks and thereby implement completely custom data handling.

See Also: DownloadHandler.ReceiveData, DownloadHandler.ReceiveContentLength, DownloadHandler.CompleteContent.

Constructors

DownloadHandlerScript	Create a DownloadHandlerScript which allocates new buffers when passing data to callbacks.

Inherited Members

Properties

data	Returns the raw bytes downloaded from the remote server, or null. (Read Only)
isDone	Returns true if this DownloadHandler has been informed by its parent UnityWebRequest that all data has been received, and this DownloadHandler has completed any necessary post-download processing. (Read Only)
text	Convenience property. Returns the bytes from data interpreted as a UTF8 string. (Read Only)

Public Methods

Dispose	Signals that this DownloadHandler is no longer being used, and should clean up any resources it is using.

Protected Methods

CompleteContent	Callback, invoked when all data has been received from the remote server.
GetData	Callback, invoked when the data property is accessed.

GetProgress	Callback, invoked when UnityWebRequest.downloadProgress is accessed.
GetText	Callback, invoked when the text property is accessed.
ReceiveContentLength	Callback, invoked with a Content-Length header is received.
ReceiveData	Callback, invoked as data is received from the remote server.

Description

A DownloadHandler subclass specialized for downloading images for use as Texture objects.

DownloadHandlerTexture stores received data in a pre-allocated Unity Texture object. It is optimized for downloading images from Web servers, and performs image decompression and decoding on a worker thread.

For use cases where you wish to download an image via HTTP and use it as a Texture within Unity, usage of this class is strongly recommended.

Properties

texture	Returns the downloaded Texture, or null. (Read Only)

Constructors

DownloadHandlerTexture	Default constructor.

Protected Methods

GetData	Called by DownloadHandler.data. Returns a copy of the downloaded image data as raw bytes.

Static Methods

GetContent	Returns the downloaded Texture, or null.

Inherited Members

Properties

data	Returns the raw bytes downloaded from the remote server, or null. (Read Only)
isDone	Returns true if this DownloadHandler has been informed by its parent UnityWebRequest that all data has been received, and this DownloadHandler has completed any necessary post-download processing. (Read Only)
text	Convenience property. Returns the bytes from data interpreted as a UTF8 string. (Read Only)

Public Methods

Dispose	Signals that this DownloadHandler is no longer being used, and should clean up any resources it is using.

Protected Methods

CompleteContent	Callback, invoked when all data has been received from the remote server.
GetProgress	Callback, invoked when UnityWebRequest.downloadProgress is accessed.
GetText	Callback, invoked when the text property is accessed.
ReceiveContentLength	Callback, invoked with a Content-Length header is received.
ReceiveData	Callback, invoked as data is received from the remote server.

Obsolete

Description

Defines global paramters for network library.

The NetworkManager has a GlobalConfig instance (NetworkManager.globalConfig) that is used to configure the network when started by the NetworkManager.

Some of the attributes of this instance can be modified from the NetworkManager inspector UI in the editor.

Obsolete

Description

Class defines network topology for host (socket opened by Networking.NetworkTransport.AddHost function). This topology defines: (1) how many connection with default config will be supported and (2) what will be special connections (connections with config different from default).

Example: p2p game with 10 people in the room and connection to chat server, in this case it will be 10 default connections and one special.

The NetworkServer and NetworkClient classed can be configured with a HostTopology instance.

Obsolete

Description

FilterLog is a utility class that controls the level of logging generated by UNET clients and servers.

Obsolete

Description

Control how verbose the network log messages are.

Obsolete

Description

Network message classes should be derived from this class. These message classes can then be sent using the various Send functions of NetworkConnection, NetworkClient and NetworkServer.

Public data fields of classes derived from MessageBase will be automatically serialized with the class. The virtual methods Serialize and Deserialize may be implemented by developers for precise control, but if they are not implemented, then implementations will be generated for them.

Note : Unity uses its own network serialization system. It doesn't support the NonSerialized attribute. Instead, use private variables.

In the example below, the methods have implementations, but if those methods were not implemented, the message would still be usable.

```
using UnityEngine;
using UnityEngine.Networking;


class SpawnMessage : MessageBase
{
    public uint netId;
    public NetworkHash128 assetId;
    public Vector3 position;
    public byte[] payload;


    // This method would be generated
    public override void Deserialize(NetworkReader reader)
    {
        netId = reader.ReadPackedUInt32();
        assetId = reader.ReadNetworkHash128();
        position = reader.ReadVector3();
        payload = reader.ReadBytesAndSize();
    }


    // This method would be generated
    public override void Serialize(NetworkWriter writer)
    {
        writer.WritePackedUInt32(netId);
        writer.Write(assetId);
```

```
        writer.Write(position);
        writer.WriteBytesFull(payload);
    }
}
```

Obsolete

Description

Container class for networking system built-in message types.

Implements interfaces:IMultipartFormSection

Description

A helper object for form sections containing generic, non-file data.

This helper object is used similarly to the WWWForm method WWWForm.AddData. It is used to define non-file form sections.

Properties

contentType	Returns the value to use in this section's Content-Type header.
fileName	Returns a string denoting the desired filename of this section on the destination server.
sectionData	Returns the raw binary data contained in this section. Will not return null or a zero-length array.
sectionName	Returns the name of this section, if any.

Constructors

MultipartFormDataSection	Raw data section, unnamed and no Content-Type header.

Implements interfaces:IMultipartFormSection

Description

A helper object for adding file uploads to multipart forms via the [IMultipartFormSection] API.

This object is very similar to the [MultipartFormDataSection] object, but all constructors additionally accept (and require) a `fileName parameter`. If you omit the `fileName` parameter, this object will provide a default filename.

Properties

contentType	Returns the value of the section's Content-Type header.
fileName	Returns a string denoting the desired filename of this section on the destination server.
sectionData	Returns the raw binary data contained in this section. Will not return null or a zero-length array.
sectionName	Returns the name of this section, if any.

Constructors

MultipartFormFileSection	Contains a named file section based on the raw bytes from data, with a custom Content-Type and file name.

Obsolete

Description

A component to synchronize Mecanim animation states for networked objects.

The animation of game objects can be networked by this component. There are two models of authority for networked movement:

If the object has authority on the client, then it should animated locally on the owning client. The animation state information will be sent from the owning client to the server, then broadcast to all of the other clients. This is common for player objects.

If the object has authority on the server, then it should be animated on the server and state information will be sent to all clients. This is common for objects not related to a specific client, such as an enemy unit.

The NetworkAnimator synchronizes the animation parameters that are checked in the inspector view. It does not automatically sychronize triggers. The function SetTrigger can by used by an object with authority to fire an animation trigger on other clients.

Inherited Members

Properties

enabled	Enabled Behaviours are Updated, disabled Behaviours are not.
isActiveAndEnabled	Has the Behaviour had active and enabled called?
gameObject	The game object this component is attached to. A component is always attached to a game object.
tag	The tag of this game object.
transform	The Transform attached to this GameObject.
runInEditMode	Allow a specific instance of a MonoBehaviour to run in edit mode (only available in the editor).
useGUILayout	Disabling this lets you skip the GUI layout phase.
hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.

Public Methods

BroadcastMessage	Calls the method named methodName on every MonoBehaviour in this game

	object or any of its children.
CompareTag	Is this game object tagged with tag ?
GetComponent	Returns the component of Type type if the game object has one attached, null if it doesn't.
GetComponentInChildren	Returns the component of Type type in the GameObject or any of its children using depth first search.
GetComponentInParent	Returns the component of Type type in the GameObject or any of its parents.
GetComponents	Returns all components of Type type in the GameObject.
GetComponentsInChildren	Returns all components of Type type in the GameObject or any of its children.
GetComponentsInParent	Returns all components of Type type in the GameObject or any of its parents.
SendMessage	Calls the method named methodName on every MonoBehaviour in this game object.
SendMessageUpwards	Calls the method named methodName on every MonoBehaviour in this game object and on every ancestor of the behaviour.
CancelInvoke	Cancels all Invoke calls on this MonoBehaviour.
Invoke	Invokes the method methodName in time seconds.
InvokeRepeating	Invokes the method methodName in time seconds, then repeatedly every repeatRate seconds.
IsInvoking	Is any invoke on methodName pending?
StartCoroutine	Starts a coroutine.
StopAllCoroutines	Stops all coroutines running on this behaviour.
StopCoroutine	Stops the first coroutine named methodName, or the coroutine stored in routine running on this behaviour.
GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.

Static Methods

print	Logs message to the Unity Console (identical to Debug.Log).
Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.

DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Messages

Awake	Awake is called when the script instance is being loaded.
FixedUpdate	This function is called every fixed framerate frame, if the MonoBehaviour is enabled.
LateUpdate	LateUpdate is called every frame, if the Behaviour is enabled.
OnAnimatorIK	Callback for setting up animation IK (inverse kinematics).
OnAnimatorMove	Callback for processing animation movements for modifying root motion.
OnApplicationFocus	Sent to all GameObjects when the player gets or loses focus.
OnApplicationPause	Sent to all GameObjects when the application pauses.
OnApplicationQuit	Sent to all game objects before the application quits.
OnAudioFilterRead	If OnAudioFilterRead is implemented, Unity will insert a custom filter into the audio DSP chain.
OnBecameInvisible	OnBecameInvisible is called when the renderer is no longer visible by any camera.
OnBecameVisible	OnBecameVisible is called when the renderer became visible by any camera.
OnCollisionEnter	OnCollisionEnter is called when this collider/rigidbody has begun touching another rigidbody/collider.
OnCollisionEnter2D	Sent when an incoming collider makes contact with this object's collider (2D physics only).

OnCollisionExit	OnCollisionExit is called when this collider/rigidbody has stopped touching another rigidbody/collider.
OnCollisionExit2D	Sent when a collider on another object stops touching this object's collider (2D physics only).
OnCollisionStay	:ref::OnCollisionStay is called once per frame for every collider/rigidbody that is touching rigidbody/collider.
OnCollisionStay2D	Sent each frame where a collider on another object is touching this object's collider (2D physics only).
OnConnectedToServer	Called on the client when you have successfully connected to a server.
OnControllerColliderHit	OnControllerColliderHit is called when the controller hits a collider while performing a Move.
OnDestroy	Destroying the attached Behaviour will result in the game or Scene receiving OnDestroy.
OnDisable	This function is called when the behaviour becomes disabled.
OnDisconnectedFromServer	Called on the client when the connection was lost or you disconnected from the server.
OnDrawGizmos	Implement OnDrawGizmos if you want to draw gizmos that are also pickable and always drawn.
OnDrawGizmosSelected	Implement OnDrawGizmosSelected to draw a gizmo if the object is selected.
OnEnable	This function is called when the object becomes enabled and active.
OnFailedToConnect	Called on the client when a connection attempt fails for some reason.
OnFailedToConnectToMasterServer	Called on clients or servers when there is a problem connecting to the MasterServer.
OnGUI	OnGUI is called for rendering and handling GUI events.
OnJointBreak	Called when a joint attached to the same game object broke.
OnJointBreak2D	Called when a Joint2D attached to the same game object breaks.
OnMasterServerEvent	Called on clients or servers when reporting events from the MasterServer.
OnMouseDown	OnMouseDown is called when the user has pressed the mouse button while over the GUIElement or Collider.
OnMouseDrag	OnMouseDrag is called when the user has clicked on a GUIElement or Collider and is still holding down the mouse.

OnMouseEnter	Called when the mouse enters the GUIElement or Collider.
OnMouseExit	Called when the mouse is not any longer over the GUIElement or Collider.
OnMouseOver	Called every frame while the mouse is over the GUIElement or Collider.
OnMouseUp	OnMouseUp is called when the user has released the mouse button.
OnMouseUpAsButton	OnMouseUpAsButton is only called when the mouse is released over the same GUIElement or Collider as it was pressed.
OnNetworkInstantiate	Called on objects which have been network instantiated with Network.Instantiate.
OnParticleCollision	OnParticleCollision is called when a particle hits a Collider.
OnParticleSystemStopped	OnParticleSystemStopped is called when all particles in the system have died, and no new particles will be born. New particles cease to be created either after Stop is called, or when the duration property of a non-looping system has been exceeded.
OnParticleTrigger	OnParticleTrigger is called when any particles in a particle system meet the conditions in the trigger module.
OnPlayerConnected	Called on the server whenever a new player has successfully connected.
OnPlayerDisconnected	Called on the server whenever a player disconnected from the server.
OnPostRender	OnPostRender is called after a camera finished rendering the Scene.
OnPreCull	OnPreCull is called before a camera culls the Scene.
OnPreRender	OnPreRender is called before a camera starts rendering the Scene.
OnRenderImage	OnRenderImage is called after all rendering is complete to render image.
OnRenderObject	OnRenderObject is called after camera has rendered the Scene.
OnSerializeNetworkView	Used to customize synchronization of variables in a script watched by a network view.
OnServerInitialized	Called on the server whenever a Network.InitializeServer was invoked and has completed.
OnTransformChildrenChanged	This function is called when the list of children of the transform of the GameObject has changed.
OnTransformParentChanged	This function is called when the parent property of the transform of the GameObject has changed.
OnTriggerEnter	OnTriggerEnter is called when the Collider other enters the trigger.

OnTriggerEnter2D	Sent when another object enters a trigger collider attached to this object (2D physics only).
OnTriggerExit	OnTriggerExit is called when the Collider other has stopped touching the trigger.
OnTriggerExit2D	Sent when another object leaves a trigger collider attached to this object (2D physics only).
OnTriggerStay	OnTriggerStay is called once per physics update for every Collider other that is touching the trigger.
OnTriggerStay2D	Sent each frame where another object is within a trigger collider attached to this object (2D physics only).
OnValidate	This function is called when the script is loaded or a value is changed in the inspector (Called in the editor only).
OnWillRenderObject	OnWillRenderObject is called for each camera if the object is visible and not a UI element.
Reset	Reset to default values.
Start	Start is called on the frame when a script is enabled just before any of the Update methods are called the first time.
Update	Update is called every frame, if the MonoBehaviour is enabled.

Obsolete

Description

Base class which should be inherited by scripts which contain networking functionality.

This is a MonoBehaviour class so scripts which need to use the networking feature should inherit this class instead of MonoBehaviour. It allows you to invoke networked actions, receive various callbacks, and automatically synchronize state from server-to-client.

The NetworkBehaviour component requires a NetworkIdentity on the game object. There can be multiple NetworkBehaviours on a single game object. For an object with sub-components in a hierarchy, the NetworkIdentity must be on the root object, and NetworkBehaviour scripts must also be on the root object.

Some of the built-in components of the networking system are derived from NetworkBehaviour, including NetworkTransport, NetworkAnimator and NetworkProximityChecker.

Inherited Members

Properties

enabled	Enabled Behaviours are Updated, disabled Behaviours are not.
isActiveAndEnabled	Has the Behaviour had active and enabled called?
gameObject	The game object this component is attached to. A component is always attached to a game object.
tag	The tag of this game object.
transform	The Transform attached to this GameObject.
runInEditMode	Allow a specific instance of a MonoBehaviour to run in edit mode (only available in the editor).
useGUILayout	Disabling this lets you skip the GUI layout phase.
hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.

Public Methods

BroadcastMessage	Calls the method named methodName on every MonoBehaviour in this game object or any of its children.
CompareTag	Is this game object tagged with tag ?

GetComponent	Returns the component of Type type if the game object has one attached, null if it doesn't.
GetComponentInChildren	Returns the component of Type type in the GameObject or any of its children using depth first search.
GetComponentInParent	Returns the component of Type type in the GameObject or any of its parents.
GetComponents	Returns all components of Type type in the GameObject.
GetComponentsInChildren	Returns all components of Type type in the GameObject or any of its children.
GetComponentsInParent	Returns all components of Type type in the GameObject or any of its parents.
SendMessage	Calls the method named methodName on every MonoBehaviour in this game object.
SendMessageUpwards	Calls the method named methodName on every MonoBehaviour in this game object and on every ancestor of the behaviour.
CancelInvoke	Cancels all Invoke calls on this MonoBehaviour.
Invoke	Invokes the method methodName in time seconds.
InvokeRepeating	Invokes the method methodName in time seconds, then repeatedly every repeatRate seconds.
IsInvoking	Is any invoke on methodName pending?
StartCoroutine	Starts a coroutine.
StopAllCoroutines	Stops all coroutines running on this behaviour.
StopCoroutine	Stops the first coroutine named methodName, or the coroutine stored in routine running on this behaviour.
GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.

Static Methods

print	Logs message to the Unity Console (identical to Debug.Log).
Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.

FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Messages

Awake	Awake is called when the script instance is being loaded.
FixedUpdate	This function is called every fixed framerate frame, if the MonoBehaviour is enabled.
LateUpdate	LateUpdate is called every frame, if the Behaviour is enabled.
OnAnimatorIK	Callback for setting up animation IK (inverse kinematics).
OnAnimatorMove	Callback for processing animation movements for modifying root motion.
OnApplicationFocus	Sent to all GameObjects when the player gets or loses focus.
OnApplicationPause	Sent to all GameObjects when the application pauses.
OnApplicationQuit	Sent to all game objects before the application quits.
OnAudioFilterRead	If OnAudioFilterRead is implemented, Unity will insert a custom filter into the audio DSP chain.
OnBecameInvisible	OnBecameInvisible is called when the renderer is no longer visible by any camera.
OnBecameVisible	OnBecameVisible is called when the renderer became visible by any camera.
OnCollisionEnter	OnCollisionEnter is called when this collider/rigidbody has begun touching another rigidbody/collider.
OnCollisionEnter2D	Sent when an incoming collider makes contact with this object's collider (2D physics only).
OnCollisionExit	OnCollisionExit is called when this collider/rigidbody has stopped touching another rigidbody/collider.
OnCollisionExit2D	Sent when a collider on another object stops touching this object's

	collider (2D physics only).
OnCollisionStay	:ref::OnCollisionStay is called once per frame for every collider/rigidbody that is touching rigidbody/collider.
OnCollisionStay2D	Sent each frame where a collider on another object is touching this object's collider (2D physics only).
OnConnectedToServer	Called on the client when you have successfully connected to a server.
OnControllerColliderHit	OnControllerColliderHit is called when the controller hits a collider while performing a Move.
OnDestroy	Destroying the attached Behaviour will result in the game or Scene receiving OnDestroy.
OnDisable	This function is called when the behaviour becomes disabled.
OnDisconnectedFromServer	Called on the client when the connection was lost or you disconnected from the server.
OnDrawGizmos	Implement OnDrawGizmos if you want to draw gizmos that are also pickable and always drawn.
OnDrawGizmosSelected	Implement OnDrawGizmosSelected to draw a gizmo if the object is selected.
OnEnable	This function is called when the object becomes enabled and active.
OnFailedToConnect	Called on the client when a connection attempt fails for some reason.
OnFailedToConnectToMasterServer	Called on clients or servers when there is a problem connecting to the MasterServer.
OnGUI	OnGUI is called for rendering and handling GUI events.
OnJointBreak	Called when a joint attached to the same game object broke.
OnJointBreak2D	Called when a Joint2D attached to the same game object breaks.
OnMasterServerEvent	Called on clients or servers when reporting events from the MasterServer.
OnMouseDown	OnMouseDown is called when the user has pressed the mouse button while over the GUIElement or Collider.
OnMouseDrag	OnMouseDrag is called when the user has clicked on a GUIElement or Collider and is still holding down the mouse.
OnMouseEnter	Called when the mouse enters the GUIElement or Collider.
OnMouseExit	Called when the mouse is not any longer over the GUIElement or Collider.

OnMouseOver	Called every frame while the mouse is over the GUIElement or Collider.
OnMouseUp	OnMouseUp is called when the user has released the mouse button.
OnMouseUpAsButton	OnMouseUpAsButton is only called when the mouse is released over the same GUIElement or Collider as it was pressed.
OnNetworkInstantiate	Called on objects which have been network instantiated with Network.Instantiate.
OnParticleCollision	OnParticleCollision is called when a particle hits a Collider.
OnParticleSystemStopped	OnParticleSystemStopped is called when all particles in the system have died, and no new particles will be born. New particles cease to be created either after Stop is called, or when the duration property of a non-looping system has been exceeded.
OnParticleTrigger	OnParticleTrigger is called when any particles in a particle system meet the conditions in the trigger module.
OnPlayerConnected	Called on the server whenever a new player has successfully connected.
OnPlayerDisconnected	Called on the server whenever a player disconnected from the server.
OnPostRender	OnPostRender is called after a camera finished rendering the Scene.
OnPreCull	OnPreCull is called before a camera culls the Scene.
OnPreRender	OnPreRender is called before a camera starts rendering the Scene.
OnRenderImage	OnRenderImage is called after all rendering is complete to render image.
OnRenderObject	OnRenderObject is called after camera has rendered the Scene.
OnSerializeNetworkView	Used to customize synchronization of variables in a script watched by a network view.
OnServerInitialized	Called on the server whenever a Network.InitializeServer was invoked and has completed.
OnTransformChildrenChanged	This function is called when the list of children of the transform of the GameObject has changed.
OnTransformParentChanged	This function is called when the parent property of the transform of the GameObject has changed.
OnTriggerEnter	OnTriggerEnter is called when the Collider other enters the trigger.
OnTriggerEnter2D	Sent when another object enters a trigger collider attached to this object (2D physics only).
OnTriggerExit	OnTriggerExit is called when the Collider other has stopped touching

	the trigger.
OnTriggerExit2D	Sent when another object leaves a trigger collider attached to this object (2D physics only).
OnTriggerStay	OnTriggerStay is called once per physics update for every Collider other that is touching the trigger.
OnTriggerStay2D	Sent each frame where another object is within a trigger collider attached to this object (2D physics only).
OnValidate	This function is called when the script is loaded or a value is changed in the inspector (Called in the editor only).
OnWillRenderObject	OnWillRenderObject is called for each camera if the object is visible and not a UI element.
Reset	Reset to default values.
Start	Start is called on the frame when a script is enabled just before any of the Update methods are called the first time.
Update	Update is called every frame, if the MonoBehaviour is enabled.

Obsolete

Description

A structure that contains data from a NetworkDiscovery server broadcast.

Obsolete

Description

This is a network client class used by the networking system. It contains a NetworkConnection that is used to connect to a network server.

The NetworkClient handle connection state, messages handlers, and connection configuration. There can be many NetworkClient instances in a process at a time, but only one that is connected to a game server (NetworkServer) that uses spawned objects.

NetworkClient has an internal update function where it handles events from the transport layer. This includes asynchronous connect events, disconnect events and incoming data from a server.

The NetworkManager has a NetworkClient instance that it uses for games that it starts, but the NetworkClient may be used by itself.

Obsolete

Description

A High level network connection. This is used for connections from client-to-server and for connection from server-to-client.

A NetworkConnection corresponds to a specific connection for a host in the transport layer. It has a connectionId that is assigned by the transport layer and passed to the Initialize function.

A NetworkClient has one NetworkConnection. A NetworkServerSimple manages multiple NetworkConnections. The NetworkServer has multiple "remote" connections and a "local" connection for the local client.

The NetworkConnection class provides message sending and handling facilities. For sending data over a network, there are methods to send message objects, byte arrays, and NetworkWriter objects. To handle data arriving from the network, handler functions can be registered for message Ids, byte arrays can be processed by HandleBytes(), and NetworkReader object can be processed by HandleReader().

NetworkConnection objects also act as observers for networked objects. When a connection is an observer of a networked object with a NetworkIdentity, then the object will be visible to corresponding client for the connection, and incremental state changes will be sent to the client.

NetworkConnection objects can "own" networked game objects. Owned objects will be destroyed on the server by default when the connection is destroyed. A connection owns the player objects created by its client, and other objects with client-authority assigned to the corresponding client.

There are many virtual functions on NetworkConnection that allow its behaviour to be customized. NetworkClient and NetworkServer can both be made to instantiate custom classes derived from NetworkConnection by setting their networkConnectionClass member variable.

Obsolete

Description

Structure used to track the number and size of packets of each packets type.

Obsolete

Description

This class holds information about which networked scripts use which QoS channels for updates.

This channel information is used to ensure that clients and servers are using compatible HLAPI script configurations.

Static Methods

ReinitializeScriptCRCs	This can be used to reinitialize the set of script CRCs.

Obsolete

Description

The NetworkDiscovery component allows Unity games to find each other on a local network. It can broadcast presence and listen for broadcasts, and optionally join matching games using the NetworkManager.

This component can run in server mode (by calling StartAsServer) where it broadcasts to other computers on the local network, or in client mode (by calling StartAsClient) where it listens for broadcasts from a server. This class should be override to receive calls from OnReceivedBroadcast.

Note : Do not use void Update() in a class that inherits from NetworkDiscovery. If needed, you must override it and call base.Update() instead.

```
using UnityEngine;
using UnityEngine.Networking;
using System.Collections;


public class OverriddenNetworkDiscovery : NetworkDiscovery
{
    public override void OnReceivedBroadcast(string fromAddress, string data)
    {
        NetworkManager.singleton.networkAddress = fromAddress;
        NetworkManager.singleton.StartClient();
    }
}
```

Inherited Members

Properties

enabled	Enabled Behaviours are Updated, disabled Behaviours are not.
isActiveAndEnabled	Has the Behaviour had active and enabled called?
gameObject	The game object this component is attached to. A component is always attached to a game object.
tag	The tag of this game object.
transform	The Transform attached to this GameObject.
runInEditMode	Allow a specific instance of a MonoBehaviour to run in edit mode (only available in the editor).

useGUILayout	Disabling this lets you skip the GUI layout phase.
hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.

Public Methods

BroadcastMessage	Calls the method named methodName on every MonoBehaviour in this game object or any of its children.
CompareTag	Is this game object tagged with tag ?
GetComponent	Returns the component of Type type if the game object has one attached, null if it doesn't.
GetComponentInChildren	Returns the component of Type type in the GameObject or any of its children using depth first search.
GetComponentInParent	Returns the component of Type type in the GameObject or any of its parents.
GetComponents	Returns all components of Type type in the GameObject.
GetComponentsInChildren	Returns all components of Type type in the GameObject or any of its children.
GetComponentsInParent	Returns all components of Type type in the GameObject or any of its parents.
SendMessage	Calls the method named methodName on every MonoBehaviour in this game object.
SendMessageUpwards	Calls the method named methodName on every MonoBehaviour in this game object and on every ancestor of the behaviour.
CancelInvoke	Cancels all Invoke calls on this MonoBehaviour.
Invoke	Invokes the method methodName in time seconds.
InvokeRepeating	Invokes the method methodName in time seconds, then repeatedly every repeatRate seconds.
IsInvoking	Is any invoke on methodName pending?
StartCoroutine	Starts a coroutine.
StopAllCoroutines	Stops all coroutines running on this behaviour.
StopCoroutine	Stops the first coroutine named methodName, or the coroutine stored in routine running on this behaviour.
GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.

Static Methods

print	Logs message to the Unity Console (identical to Debug.Log).
Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Messages

Awake	Awake is called when the script instance is being loaded.
FixedUpdate	This function is called every fixed framerate frame, if the MonoBehaviour is enabled.
LateUpdate	LateUpdate is called every frame, if the Behaviour is enabled.
OnAnimatorIK	Callback for setting up animation IK (inverse kinematics).
OnAnimatorMove	Callback for processing animation movements for modifying root motion.
OnApplicationFocus	Sent to all GameObjects when the player gets or loses focus.
OnApplicationPause	Sent to all GameObjects when the application pauses.
OnApplicationQuit	Sent to all game objects before the application quits.
OnAudioFilterRead	If OnAudioFilterRead is implemented, Unity will insert a custom filter into the audio DSP chain.
OnBecameInvisible	OnBecameInvisible is called when the renderer is no longer visible by

	any camera.
OnBecameVisible	OnBecameVisible is called when the renderer became visible by any camera.
OnCollisionEnter	OnCollisionEnter is called when this collider/rigidbody has begun touching another rigidbody/collider.
OnCollisionEnter2D	Sent when an incoming collider makes contact with this object's collider (2D physics only).
OnCollisionExit	OnCollisionExit is called when this collider/rigidbody has stopped touching another rigidbody/collider.
OnCollisionExit2D	Sent when a collider on another object stops touching this object's collider (2D physics only).
OnCollisionStay	:ref::OnCollisionStay is called once per frame for every collider/rigidbody that is touching rigidbody/collider.
OnCollisionStay2D	Sent each frame where a collider on another object is touching this object's collider (2D physics only).
OnConnectedToServer	Called on the client when you have successfully connected to a server.
OnControllerColliderHit	OnControllerColliderHit is called when the controller hits a collider while performing a Move.
OnDestroy	Destroying the attached Behaviour will result in the game or Scene receiving OnDestroy.
OnDisable	This function is called when the behaviour becomes disabled.
OnDisconnectedFromServer	Called on the client when the connection was lost or you disconnected from the server.
OnDrawGizmos	Implement OnDrawGizmos if you want to draw gizmos that are also pickable and always drawn.
OnDrawGizmosSelected	Implement OnDrawGizmosSelected to draw a gizmo if the object is selected.
OnEnable	This function is called when the object becomes enabled and active.
OnFailedToConnect	Called on the client when a connection attempt fails for some reason.
OnFailedToConnectToMasterServer	Called on clients or servers when there is a problem connecting to the MasterServer.
OnGUI	OnGUI is called for rendering and handling GUI events.
OnJointBreak	Called when a joint attached to the same game object broke.

OnJointBreak2D	Called when a Joint2D attached to the same game object breaks.
OnMasterServerEvent	Called on clients or servers when reporting events from the MasterServer.
OnMouseDown	OnMouseDown is called when the user has pressed the mouse button while over the GUIElement or Collider.
OnMouseDrag	OnMouseDrag is called when the user has clicked on a GUIElement or Collider and is still holding down the mouse.
OnMouseEnter	Called when the mouse enters the GUIElement or Collider.
OnMouseExit	Called when the mouse is not any longer over the GUIElement or Collider.
OnMouseOver	Called every frame while the mouse is over the GUIElement or Collider.
OnMouseUp	OnMouseUp is called when the user has released the mouse button.
OnMouseUpAsButton	OnMouseUpAsButton is only called when the mouse is released over the same GUIElement or Collider as it was pressed.
OnNetworkInstantiate	Called on objects which have been network instantiated with Network.Instantiate.
OnParticleCollision	OnParticleCollision is called when a particle hits a Collider.
OnParticleSystemStopped	OnParticleSystemStopped is called when all particles in the system have died, and no new particles will be born. New particles cease to be created either after Stop is called, or when the duration property of a non-looping system has been exceeded.
OnParticleTrigger	OnParticleTrigger is called when any particles in a particle system meet the conditions in the trigger module.
OnPlayerConnected	Called on the server whenever a new player has successfully connected.
OnPlayerDisconnected	Called on the server whenever a player disconnected from the server.
OnPostRender	OnPostRender is called after a camera finished rendering the Scene.
OnPreCull	OnPreCull is called before a camera culls the Scene.
OnPreRender	OnPreRender is called before a camera starts rendering the Scene.
OnRenderImage	OnRenderImage is called after all rendering is complete to render image.
OnRenderObject	OnRenderObject is called after camera has rendered the Scene.
OnSerializeNetworkView	Used to customize synchronization of variables in a script watched by a network view.

OnServerInitialized	Called on the server whenever a Network.InitializeServer was invoked and has completed.
OnTransformChildrenChanged	This function is called when the list of children of the transform of the GameObject has changed.
OnTransformParentChanged	This function is called when the parent property of the transform of the GameObject has changed.
OnTriggerEnter	OnTriggerEnter is called when the Collider other enters the trigger.
OnTriggerEnter2D	Sent when another object enters a trigger collider attached to this object (2D physics only).
OnTriggerExit	OnTriggerExit is called when the Collider other has stopped touching the trigger.
OnTriggerExit2D	Sent when another object leaves a trigger collider attached to this object (2D physics only).
OnTriggerStay	OnTriggerStay is called once per physics update for every Collider other that is touching the trigger.
OnTriggerStay2D	Sent each frame where another object is within a trigger collider attached to this object (2D physics only).
OnValidate	This function is called when the script is loaded or a value is changed in the inspector (Called in the editor only).
OnWillRenderObject	OnWillRenderObject is called for each camera if the object is visible and not a UI element.
Reset	Reset to default values.
Start	Start is called on the frame when a script is enabled just before any of the Update methods are called the first time.
Update	Update is called every frame, if the MonoBehaviour is enabled.

Obsolete

Description

A 128 bit number used to represent assets in a networking context.

Obsolete

Description

The NetworkIdentity identifies objects across the network, between server and clients. Its primary data is a NetworkInstanceId which is allocated by the server and then set on clients. This is used in network communications to be able to lookup game objects on different machines.

The NetworkIdentity is used to synchronize information in the object with the network. Only the server should create instances of objects which have NetworkIdentity as otherwise they will not be properly connected to the system.

For complex objects with a hierarchy of subcomponents, the NetworkIdentity must be on the root of the hierarchy. It is not supported to have multiple NetworkIdentity components on subcomponents of a hierarchy.

NetworkBehaviour scripts require a NetworkIdentity on the game object to be able to function.

The NetworkIdentity manages the dirty state of the NetworkBehaviours of the object. When it discovers that NetworkBehaviours are dirty, it causes an update packet to be created and sent to clients.

The flow for serialization updates managed by the NetworkIdentity is:

* Each NetworkBehaviour has a dirty mask. This mask is available inside OnSerialize as syncVarDirtyBits

* Each SyncVar in a NetworkBehaviour script is assigned a bit in the dirty mask.

* Changing the value of SyncVars causes the bit for that SyncVar to be set in the dirty mask

* Alternatively, calling SetDirtyBit() writes directly to the dirty mask

* NetworkIdentity objects are checked on the server as part of it's update loop

* If any NetworkBehaviours on a NetworkIdentity are dirty, then an UpdateVars packet is created for that object

* The UpdateVars packet is populated by calling OnSerialize on each NetworkBehaviour on the object

* NetworkBehaviours that are NOT dirty write a zero to the packet for their dirty bits

* NetworkBehaviours that are dirty write their dirty mask, then the values for the SyncVars that have changed

* If OnSerialize returns true for a NetworkBehaviour, the dirty mask is reset for that NetworkBehaviour, so it will not send again until its value changes.

* The UpdateVars packet is sent to ready clients that are observing the object

On the client:

* an UpdateVars packet is received for an object

* The OnDeserialize function is called for each NetworkBehaviour script on the object

* Each NetworkBehaviour script on the object reads a dirty mask.

* If the dirty mask for a NetworkBehaviour is zero, the OnDeserialize functions returns without reading any more

* If the dirty mask is non-zero value, then the OnDeserialize function reads the values for the SyncVars that correspond to the dirty bits that are set

* If there are SyncVar hook functions, those are invoked with the value read from the stream.

Inherited Members

Properties

enabled	Enabled Behaviours are Updated, disabled Behaviours are not.
isActiveAndEnabled	Has the Behaviour had active and enabled called?
gameObject	The game object this component is attached to. A component is always attached to a game object.
tag	The tag of this game object.
transform	The Transform attached to this GameObject.
runInEditMode	Allow a specific instance of a MonoBehaviour to run in edit mode (only available in the editor).
useGUILayout	Disabling this lets you skip the GUI layout phase.
hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.

Public Methods

BroadcastMessage	Calls the method named methodName on every MonoBehaviour in this game object or any of its children.
CompareTag	Is this game object tagged with tag ?
GetComponent	Returns the component of Type type if the game object has one attached, null if it doesn't.
GetComponentInChildren	Returns the component of Type type in the GameObject or any of its children

	using depth first search.
GetComponentInParent	Returns the component of Type type in the GameObject or any of its parents.
GetComponents	Returns all components of Type type in the GameObject.
GetComponentsInChildren	Returns all components of Type type in the GameObject or any of its children.
GetComponentsInParent	Returns all components of Type type in the GameObject or any of its parents.
SendMessage	Calls the method named methodName on every MonoBehaviour in this game object.
SendMessageUpwards	Calls the method named methodName on every MonoBehaviour in this game object and on every ancestor of the behaviour.
CancelInvoke	Cancels all Invoke calls on this MonoBehaviour.
Invoke	Invokes the method methodName in time seconds.
InvokeRepeating	Invokes the method methodName in time seconds, then repeatedly every repeatRate seconds.
IsInvoking	Is any invoke on methodName pending?
StartCoroutine	Starts a coroutine.
StopAllCoroutines	Stops all coroutines running on this behaviour.
StopCoroutine	Stops the first coroutine named methodName, or the coroutine stored in routine running on this behaviour.
GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.

Static Methods

print	Logs message to the Unity Console (identical to Debug.Log).
Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Messages

Awake	Awake is called when the script instance is being loaded.
FixedUpdate	This function is called every fixed framerate frame, if the MonoBehaviour is enabled.
LateUpdate	LateUpdate is called every frame, if the Behaviour is enabled.
OnAnimatorIK	Callback for setting up animation IK (inverse kinematics).
OnAnimatorMove	Callback for processing animation movements for modifying root motion.
OnApplicationFocus	Sent to all GameObjects when the player gets or loses focus.
OnApplicationPause	Sent to all GameObjects when the application pauses.
OnApplicationQuit	Sent to all game objects before the application quits.
OnAudioFilterRead	If OnAudioFilterRead is implemented, Unity will insert a custom filter into the audio DSP chain.
OnBecameInvisible	OnBecameInvisible is called when the renderer is no longer visible by any camera.
OnBecameVisible	OnBecameVisible is called when the renderer became visible by any camera.
OnCollisionEnter	OnCollisionEnter is called when this collider/rigidbody has begun touching another rigidbody/collider.
OnCollisionEnter2D	Sent when an incoming collider makes contact with this object's collider (2D physics only).
OnCollisionExit	OnCollisionExit is called when this collider/rigidbody has stopped touching another rigidbody/collider.
OnCollisionExit2D	Sent when a collider on another object stops touching this object's collider (2D physics only).
OnCollisionStay	:ref::OnCollisionStay is called once per frame for every collider/rigidbody that is touching rigidbody/collider.

OnCollisionStay2D	Sent each frame where a collider on another object is touching this object's collider (2D physics only).
OnConnectedToServer	Called on the client when you have successfully connected to a server.
OnControllerColliderHit	OnControllerColliderHit is called when the controller hits a collider while performing a Move.
OnDestroy	Destroying the attached Behaviour will result in the game or Scene receiving OnDestroy.
OnDisable	This function is called when the behaviour becomes disabled.
OnDisconnectedFromServer	Called on the client when the connection was lost or you disconnected from the server.
OnDrawGizmos	Implement OnDrawGizmos if you want to draw gizmos that are also pickable and always drawn.
OnDrawGizmosSelected	Implement OnDrawGizmosSelected to draw a gizmo if the object is selected.
OnEnable	This function is called when the object becomes enabled and active.
OnFailedToConnect	Called on the client when a connection attempt fails for some reason.
OnFailedToConnectToMasterServer	Called on clients or servers when there is a problem connecting to the MasterServer.
OnGUI	OnGUI is called for rendering and handling GUI events.
OnJointBreak	Called when a joint attached to the same game object broke.
OnJointBreak2D	Called when a Joint2D attached to the same game object breaks.
OnMasterServerEvent	Called on clients or servers when reporting events from the MasterServer.
OnMouseDown	OnMouseDown is called when the user has pressed the mouse button while over the GUIElement or Collider.
OnMouseDrag	OnMouseDrag is called when the user has clicked on a GUIElement or Collider and is still holding down the mouse.
OnMouseEnter	Called when the mouse enters the GUIElement or Collider.
OnMouseExit	Called when the mouse is not any longer over the GUIElement or Collider.
OnMouseOver	Called every frame while the mouse is over the GUIElement or Collider.
OnMouseUp	OnMouseUp is called when the user has released the mouse button.

OnMouseUpAsButton	OnMouseUpAsButton is only called when the mouse is released over the same GUIElement or Collider as it was pressed.
OnNetworkInstantiate	Called on objects which have been network instantiated with Network.Instantiate.
OnParticleCollision	OnParticleCollision is called when a particle hits a Collider.
OnParticleSystemStopped	OnParticleSystemStopped is called when all particles in the system have died, and no new particles will be born. New particles cease to be created either after Stop is called, or when the duration property of a non-looping system has been exceeded.
OnParticleTrigger	OnParticleTrigger is called when any particles in a particle system meet the conditions in the trigger module.
OnPlayerConnected	Called on the server whenever a new player has successfully connected.
OnPlayerDisconnected	Called on the server whenever a player disconnected from the server.
OnPostRender	OnPostRender is called after a camera finished rendering the Scene.
OnPreCull	OnPreCull is called before a camera culls the Scene.
OnPreRender	OnPreRender is called before a camera starts rendering the Scene.
OnRenderImage	OnRenderImage is called after all rendering is complete to render image.
OnRenderObject	OnRenderObject is called after camera has rendered the Scene.
OnSerializeNetworkView	Used to customize synchronization of variables in a script watched by a network view.
OnServerInitialized	Called on the server whenever a Network.InitializeServer was invoked and has completed.
OnTransformChildrenChanged	This function is called when the list of children of the transform of the GameObject has changed.
OnTransformParentChanged	This function is called when the parent property of the transform of the GameObject has changed.
OnTriggerEnter	OnTriggerEnter is called when the Collider other enters the trigger.
OnTriggerEnter2D	Sent when another object enters a trigger collider attached to this object (2D physics only).
OnTriggerExit	OnTriggerExit is called when the Collider other has stopped touching the trigger.
OnTriggerExit2D	Sent when another object leaves a trigger collider attached to this object (2D physics only).

OnTriggerStay	OnTriggerStay is called once per physics update for every Collider other that is touching the trigger.
OnTriggerStay2D	Sent each frame where another object is within a trigger collider attached to this object (2D physics only).
OnValidate	This function is called when the script is loaded or a value is changed in the inspector (Called in the editor only).
OnWillRenderObject	OnWillRenderObject is called for each camera if the object is visible and not a UI element.
Reset	Reset to default values.
Start	Start is called on the frame when a script is enabled just before any of the Update methods are called the first time.
Update	Update is called every frame, if the MonoBehaviour is enabled.

Obsolete

Description

This is used to identify networked objects across all participants of a network. It is assigned at runtime by the server when an object is spawned.

Obsolete

Description

This is a specialized NetworkManager that includes a networked lobby.

The lobby has slots that track the joined players, and a maximum player count that is enforced. It requires that the NetworkLobbyPlayer component be on the lobby player objects.

NetworkLobbyManager is derived from NetworkManager, and so it implements many of the virtual functions provided by the NetworkManager class. To avoid accidentally replacing functionality of the NetworkLobbyManager, there are new virtual functions on the NetworkLobbyManager that begin with "OnLobby". These should be used on classes derived from NetworkLobbyManager instead of the virtual functions on NetworkManager.

The OnLobby*() functions have empty implementations on the NetworkLobbyManager base class, so the base class functions do not have to be called.

Inherited Members

Properties

enabled	Enabled Behaviours are Updated, disabled Behaviours are not.
isActiveAndEnabled	Has the Behaviour had active and enabled called?
gameObject	The game object this component is attached to. A component is always attached to a game object.
tag	The tag of this game object.
transform	The Transform attached to this GameObject.
runInEditMode	Allow a specific instance of a MonoBehaviour to run in edit mode (only available in the editor).
useGUILayout	Disabling this lets you skip the GUI layout phase.
hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.

Public Methods

BroadcastMessage	Calls the method named methodName on every MonoBehaviour in this game object or any of its children.
CompareTag	Is this game object tagged with tag ?

GetComponent	Returns the component of Type type if the game object has one attached, null if it doesn't.
GetComponentInChildren	Returns the component of Type type in the GameObject or any of its children using depth first search.
GetComponentInParent	Returns the component of Type type in the GameObject or any of its parents.
GetComponents	Returns all components of Type type in the GameObject.
GetComponentsInChildren	Returns all components of Type type in the GameObject or any of its children.
GetComponentsInParent	Returns all components of Type type in the GameObject or any of its parents.
SendMessage	Calls the method named methodName on every MonoBehaviour in this game object.
SendMessageUpwards	Calls the method named methodName on every MonoBehaviour in this game object and on every ancestor of the behaviour.
CancelInvoke	Cancels all Invoke calls on this MonoBehaviour.
Invoke	Invokes the method methodName in time seconds.
InvokeRepeating	Invokes the method methodName in time seconds, then repeatedly every repeatRate seconds.
IsInvoking	Is any invoke on methodName pending?
StartCoroutine	Starts a coroutine.
StopAllCoroutines	Stops all coroutines running on this behaviour.
StopCoroutine	Stops the first coroutine named methodName, or the coroutine stored in routine running on this behaviour.
GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.

Static Methods

print	Logs message to the Unity Console (identical to Debug.Log).
Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.

FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Messages

Awake	Awake is called when the script instance is being loaded.
FixedUpdate	This function is called every fixed framerate frame, if the MonoBehaviour is enabled.
LateUpdate	LateUpdate is called every frame, if the Behaviour is enabled.
OnAnimatorIK	Callback for setting up animation IK (inverse kinematics).
OnAnimatorMove	Callback for processing animation movements for modifying root motion.
OnApplicationFocus	Sent to all GameObjects when the player gets or loses focus.
OnApplicationPause	Sent to all GameObjects when the application pauses.
OnApplicationQuit	Sent to all game objects before the application quits.
OnAudioFilterRead	If OnAudioFilterRead is implemented, Unity will insert a custom filter into the audio DSP chain.
OnBecameInvisible	OnBecameInvisible is called when the renderer is no longer visible by any camera.
OnBecameVisible	OnBecameVisible is called when the renderer became visible by any camera.
OnCollisionEnter	OnCollisionEnter is called when this collider/rigidbody has begun touching another rigidbody/collider.
OnCollisionEnter2D	Sent when an incoming collider makes contact with this object's collider (2D physics only).
OnCollisionExit	OnCollisionExit is called when this collider/rigidbody has stopped touching another rigidbody/collider.

OnCollisionExit2D	Sent when a collider on another object stops touching this object's collider (2D physics only).
OnCollisionStay	:ref::OnCollisionStay is called once per frame for every collider/rigidbody that is touching rigidbody/collider.
OnCollisionStay2D	Sent each frame where a collider on another object is touching this object's collider (2D physics only).
OnConnectedToServer	Called on the client when you have successfully connected to a server.
OnControllerColliderHit	OnControllerColliderHit is called when the controller hits a collider while performing a Move.
OnDestroy	Destroying the attached Behaviour will result in the game or Scene receiving OnDestroy.
OnDisable	This function is called when the behaviour becomes disabled.
OnDisconnectedFromServer	Called on the client when the connection was lost or you disconnected from the server.
OnDrawGizmos	Implement OnDrawGizmos if you want to draw gizmos that are also pickable and always drawn.
OnDrawGizmosSelected	Implement OnDrawGizmosSelected to draw a gizmo if the object is selected.
OnEnable	This function is called when the object becomes enabled and active.
OnFailedToConnect	Called on the client when a connection attempt fails for some reason.
OnFailedToConnectToMasterServer	Called on clients or servers when there is a problem connecting to the MasterServer.
OnGUI	OnGUI is called for rendering and handling GUI events.
OnJointBreak	Called when a joint attached to the same game object broke.
OnJointBreak2D	Called when a Joint2D attached to the same game object breaks.
OnMasterServerEvent	Called on clients or servers when reporting events from the MasterServer.
OnMouseDown	OnMouseDown is called when the user has pressed the mouse button while over the GUIElement or Collider.
OnMouseDrag	OnMouseDrag is called when the user has clicked on a GUIElement or Collider and is still holding down the mouse.
OnMouseEnter	Called when the mouse enters the GUIElement or Collider.
OnMouseExit	Called when the mouse is not any longer over the GUIElement or

	Collider.
OnMouseOver	Called every frame while the mouse is over the GUIElement or Collider.
OnMouseUp	OnMouseUp is called when the user has released the mouse button.
OnMouseUpAsButton	OnMouseUpAsButton is only called when the mouse is released over the same GUIElement or Collider as it was pressed.
OnNetworkInstantiate	Called on objects which have been network instantiated with Network.Instantiate.
OnParticleCollision	OnParticleCollision is called when a particle hits a Collider.
OnParticleSystemStopped	OnParticleSystemStopped is called when all particles in the system have died, and no new particles will be born. New particles cease to be created either after Stop is called, or when the duration property of a non-looping system has been exceeded.
OnParticleTrigger	OnParticleTrigger is called when any particles in a particle system meet the conditions in the trigger module.
OnPlayerConnected	Called on the server whenever a new player has successfully connected.
OnPlayerDisconnected	Called on the server whenever a player disconnected from the server.
OnPostRender	OnPostRender is called after a camera finished rendering the Scene.
OnPreCull	OnPreCull is called before a camera culls the Scene.
OnPreRender	OnPreRender is called before a camera starts rendering the Scene.
OnRenderImage	OnRenderImage is called after all rendering is complete to render image.
OnRenderObject	OnRenderObject is called after camera has rendered the Scene.
OnSerializeNetworkView	Used to customize synchronization of variables in a script watched by a network view.
OnServerInitialized	Called on the server whenever a Network.InitializeServer was invoked and has completed.
OnTransformChildrenChanged	This function is called when the list of children of the transform of the GameObject has changed.
OnTransformParentChanged	This function is called when the parent property of the transform of the GameObject has changed.
OnTriggerEnter	OnTriggerEnter is called when the Collider other enters the trigger.
OnTriggerEnter2D	Sent when another object enters a trigger collider attached to this object (2D physics only).

OnTriggerExit	OnTriggerExit is called when the Collider other has stopped touching the trigger.
OnTriggerExit2D	Sent when another object leaves a trigger collider attached to this object (2D physics only).
OnTriggerStay	OnTriggerStay is called once per physics update for every Collider other that is touching the trigger.
OnTriggerStay2D	Sent each frame where another object is within a trigger collider attached to this object (2D physics only).
OnValidate	This function is called when the script is loaded or a value is changed in the inspector (Called in the editor only).
OnWillRenderObject	OnWillRenderObject is called for each camera if the object is visible and not a UI element.
Reset	Reset to default values.
Start	Start is called on the frame when a script is enabled just before any of the Update methods are called the first time.
Update	Update is called every frame, if the MonoBehaviour is enabled.

Obsolete

Description

This component works in conjunction with the NetworkLobbyManager to make up the multiplayer lobby system.

The LobbyPrefab object of the NetworkLobbyManager must have this component on it. This component holds basic lobby player data required for the lobby to function. Game specific data for lobby players can be put in other components on the LobbyPrefab or in scripts derived from NetworkLobbyPlayer.

Inherited Members

Properties

enabled	Enabled Behaviours are Updated, disabled Behaviours are not.
isActiveAndEnabled	Has the Behaviour had active and enabled called?
gameObject	The game object this component is attached to. A component is always attached to a game object.
tag	The tag of this game object.
transform	The Transform attached to this GameObject.
runInEditMode	Allow a specific instance of a MonoBehaviour to run in edit mode (only available in the editor).
useGUILayout	Disabling this lets you skip the GUI layout phase.
hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.

Public Methods

BroadcastMessage	Calls the method named methodName on every MonoBehaviour in this game object or any of its children.
CompareTag	Is this game object tagged with tag ?
GetComponent	Returns the component of Type type if the game object has one attached, null if it doesn't.
GetComponentInChildren	Returns the component of Type type in the GameObject or any of its children using depth first search.
GetComponentInParent	Returns the component of Type type in the GameObject or any of its parents.

GetComponents	Returns all components of Type type in the GameObject.
GetComponentsInChildren	Returns all components of Type type in the GameObject or any of its children.
GetComponentsInParent	Returns all components of Type type in the GameObject or any of its parents.
SendMessage	Calls the method named methodName on every MonoBehaviour in this game object.
SendMessageUpwards	Calls the method named methodName on every MonoBehaviour in this game object and on every ancestor of the behaviour.
CancelInvoke	Cancels all Invoke calls on this MonoBehaviour.
Invoke	Invokes the method methodName in time seconds.
InvokeRepeating	Invokes the method methodName in time seconds, then repeatedly every repeatRate seconds.
IsInvoking	Is any invoke on methodName pending?
StartCoroutine	Starts a coroutine.
StopAllCoroutines	Stops all coroutines running on this behaviour.
StopCoroutine	Stops the first coroutine named methodName, or the coroutine stored in routine running on this behaviour.
GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.

Static Methods

print	Logs message to the Unity Console (identical to Debug.Log).
Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Messages

Awake	Awake is called when the script instance is being loaded.
FixedUpdate	This function is called every fixed framerate frame, if the MonoBehaviour is enabled.
LateUpdate	LateUpdate is called every frame, if the Behaviour is enabled.
OnAnimatorIK	Callback for setting up animation IK (inverse kinematics).
OnAnimatorMove	Callback for processing animation movements for modifying root motion.
OnApplicationFocus	Sent to all GameObjects when the player gets or loses focus.
OnApplicationPause	Sent to all GameObjects when the application pauses.
OnApplicationQuit	Sent to all game objects before the application quits.
OnAudioFilterRead	If OnAudioFilterRead is implemented, Unity will insert a custom filter into the audio DSP chain.
OnBecameInvisible	OnBecameInvisible is called when the renderer is no longer visible by any camera.
OnBecameVisible	OnBecameVisible is called when the renderer became visible by any camera.
OnCollisionEnter	OnCollisionEnter is called when this collider/rigidbody has begun touching another rigidbody/collider.
OnCollisionEnter2D	Sent when an incoming collider makes contact with this object's collider (2D physics only).
OnCollisionExit	OnCollisionExit is called when this collider/rigidbody has stopped touching another rigidbody/collider.
OnCollisionExit2D	Sent when a collider on another object stops touching this object's collider (2D physics only).
OnCollisionStay	:ref::OnCollisionStay is called once per frame for every collider/rigidbody that is touching rigidbody/collider.
OnCollisionStay2D	Sent each frame where a collider on another object is touching this object's collider (2D physics only).

OnConnectedToServer	Called on the client when you have successfully connected to a server.
OnControllerColliderHit	OnControllerColliderHit is called when the controller hits a collider while performing a Move.
OnDestroy	Destroying the attached Behaviour will result in the game or Scene receiving OnDestroy.
OnDisable	This function is called when the behaviour becomes disabled.
OnDisconnectedFromServer	Called on the client when the connection was lost or you disconnected from the server.
OnDrawGizmos	Implement OnDrawGizmos if you want to draw gizmos that are also pickable and always drawn.
OnDrawGizmosSelected	Implement OnDrawGizmosSelected to draw a gizmo if the object is selected.
OnEnable	This function is called when the object becomes enabled and active.
OnFailedToConnect	Called on the client when a connection attempt fails for some reason.
OnFailedToConnectToMasterServer	Called on clients or servers when there is a problem connecting to the MasterServer.
OnGUI	OnGUI is called for rendering and handling GUI events.
OnJointBreak	Called when a joint attached to the same game object broke.
OnJointBreak2D	Called when a Joint2D attached to the same game object breaks.
OnMasterServerEvent	Called on clients or servers when reporting events from the MasterServer.
OnMouseDown	OnMouseDown is called when the user has pressed the mouse button while over the GUIElement or Collider.
OnMouseDrag	OnMouseDrag is called when the user has clicked on a GUIElement or Collider and is still holding down the mouse.
OnMouseEnter	Called when the mouse enters the GUIElement or Collider.
OnMouseExit	Called when the mouse is not any longer over the GUIElement or Collider.
OnMouseOver	Called every frame while the mouse is over the GUIElement or Collider.
OnMouseUp	OnMouseUp is called when the user has released the mouse button.
OnMouseUpAsButton	OnMouseUpAsButton is only called when the mouse is released over the same GUIElement or Collider as it was pressed.

OnNetworkInstantiate	Called on objects which have been network instantiated with Network.Instantiate.
OnParticleCollision	OnParticleCollision is called when a particle hits a Collider.
OnParticleSystemStopped	OnParticleSystemStopped is called when all particles in the system have died, and no new particles will be born. New particles cease to be created either after Stop is called, or when the duration property of a non-looping system has been exceeded.
OnParticleTrigger	OnParticleTrigger is called when any particles in a particle system meet the conditions in the trigger module.
OnPlayerConnected	Called on the server whenever a new player has successfully connected.
OnPlayerDisconnected	Called on the server whenever a player disconnected from the server.
OnPostRender	OnPostRender is called after a camera finished rendering the Scene.
OnPreCull	OnPreCull is called before a camera culls the Scene.
OnPreRender	OnPreRender is called before a camera starts rendering the Scene.
OnRenderImage	OnRenderImage is called after all rendering is complete to render image.
OnRenderObject	OnRenderObject is called after camera has rendered the Scene.
OnSerializeNetworkView	Used to customize synchronization of variables in a script watched by a network view.
OnServerInitialized	Called on the server whenever a Network.InitializeServer was invoked and has completed.
OnTransformChildrenChanged	This function is called when the list of children of the transform of the GameObject has changed.
OnTransformParentChanged	This function is called when the parent property of the transform of the GameObject has changed.
OnTriggerEnter	OnTriggerEnter is called when the Collider other enters the trigger.
OnTriggerEnter2D	Sent when another object enters a trigger collider attached to this object (2D physics only).
OnTriggerExit	OnTriggerExit is called when the Collider other has stopped touching the trigger.
OnTriggerExit2D	Sent when another object leaves a trigger collider attached to this object (2D physics only).
OnTriggerStay	OnTriggerStay is called once per physics update for every Collider other that is touching the trigger.

OnTriggerStay2D	Sent each frame where another object is within a trigger collider attached to this object (2D physics only).
OnValidate	This function is called when the script is loaded or a value is changed in the inspector (Called in the editor only).
OnWillRenderObject	OnWillRenderObject is called for each camera if the object is visible and not a UI element.
Reset	Reset to default values.
Start	Start is called on the frame when a script is enabled just before any of the Update methods are called the first time.
Update	Update is called every frame, if the MonoBehaviour is enabled.

Obsolete

Description

The NetworkManager is a convenience class for the HLAPI for managing networking systems.

For simple network applications the NetworkManager can be used to control the HLAPI. It provides simple ways to start and stop client and servers, to manage Scenes, and has virtual functions that user code can use to implement handlers for network events. The NetworkManager deals with one client at a time. The example below shows a minimal network setup.

```
using UnityEngine;
using UnityEngine.Networking;


public class Manager : NetworkManager
{
    public override void OnServerConnect(NetworkConnection conn)
    {
        Debug.Log("OnPlayerConnected");
    }
}
```

Inherited Members

Properties

enabled	Enabled Behaviours are Updated, disabled Behaviours are not.
isActiveAndEnabled	Has the Behaviour had active and enabled called?
gameObject	The game object this component is attached to. A component is always attached to a game object.
tag	The tag of this game object.
transform	The Transform attached to this GameObject.
runInEditMode	Allow a specific instance of a MonoBehaviour to run in edit mode (only available in the editor).
useGUILayout	Disabling this lets you skip the GUI layout phase.
hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.

Public Methods

BroadcastMessage	Calls the method named methodName on every MonoBehaviour in this game object or any of its children.
CompareTag	Is this game object tagged with tag ?
GetComponent	Returns the component of Type type if the game object has one attached, null if it doesn't.
GetComponentInChildren	Returns the component of Type type in the GameObject or any of its children using depth first search.
GetComponentInParent	Returns the component of Type type in the GameObject or any of its parents.
GetComponents	Returns all components of Type type in the GameObject.
GetComponentsInChildren	Returns all components of Type type in the GameObject or any of its children.
GetComponentsInParent	Returns all components of Type type in the GameObject or any of its parents.
SendMessage	Calls the method named methodName on every MonoBehaviour in this game object.
SendMessageUpwards	Calls the method named methodName on every MonoBehaviour in this game object and on every ancestor of the behaviour.
CancelInvoke	Cancels all Invoke calls on this MonoBehaviour.
Invoke	Invokes the method methodName in time seconds.
InvokeRepeating	Invokes the method methodName in time seconds, then repeatedly every repeatRate seconds.
IsInvoking	Is any invoke on methodName pending?
StartCoroutine	Starts a coroutine.
StopAllCoroutines	Stops all coroutines running on this behaviour.
StopCoroutine	Stops the first coroutine named methodName, or the coroutine stored in routine running on this behaviour.
GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.

Static Methods

print	Logs message to the Unity Console (identical to Debug.Log).

Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Messages

Awake	Awake is called when the script instance is being loaded.
FixedUpdate	This function is called every fixed framerate frame, if the MonoBehaviour is enabled.
LateUpdate	LateUpdate is called every frame, if the Behaviour is enabled.
OnAnimatorIK	Callback for setting up animation IK (inverse kinematics).
OnAnimatorMove	Callback for processing animation movements for modifying root motion.
OnApplicationFocus	Sent to all GameObjects when the player gets or loses focus.
OnApplicationPause	Sent to all GameObjects when the application pauses.
OnApplicationQuit	Sent to all game objects before the application quits.
OnAudioFilterRead	If OnAudioFilterRead is implemented, Unity will insert a custom filter into the audio DSP chain.
OnBecameInvisible	OnBecameInvisible is called when the renderer is no longer visible by any camera.
OnBecameVisible	OnBecameVisible is called when the renderer became visible by any camera.
OnCollisionEnter	OnCollisionEnter is called when this collider/rigidbody has begun

	touching another rigidbody/collider.
OnCollisionEnter2D	Sent when an incoming collider makes contact with this object's collider (2D physics only).
OnCollisionExit	OnCollisionExit is called when this collider/rigidbody has stopped touching another rigidbody/collider.
OnCollisionExit2D	Sent when a collider on another object stops touching this object's collider (2D physics only).
OnCollisionStay	:ref::OnCollisionStay is called once per frame for every collider/rigidbody that is touching rigidbody/collider.
OnCollisionStay2D	Sent each frame where a collider on another object is touching this object's collider (2D physics only).
OnConnectedToServer	Called on the client when you have successfully connected to a server.
OnControllerColliderHit	OnControllerColliderHit is called when the controller hits a collider while performing a Move.
OnDestroy	Destroying the attached Behaviour will result in the game or Scene receiving OnDestroy.
OnDisable	This function is called when the behaviour becomes disabled.
OnDisconnectedFromServer	Called on the client when the connection was lost or you disconnected from the server.
OnDrawGizmos	Implement OnDrawGizmos if you want to draw gizmos that are also pickable and always drawn.
OnDrawGizmosSelected	Implement OnDrawGizmosSelected to draw a gizmo if the object is selected.
OnEnable	This function is called when the object becomes enabled and active.
OnFailedToConnect	Called on the client when a connection attempt fails for some reason.
OnFailedToConnectToMasterServer	Called on clients or servers when there is a problem connecting to the MasterServer.
OnGUI	OnGUI is called for rendering and handling GUI events.
OnJointBreak	Called when a joint attached to the same game object broke.
OnJointBreak2D	Called when a Joint2D attached to the same game object breaks.
OnMasterServerEvent	Called on clients or servers when reporting events from the MasterServer.
OnMouseDown	OnMouseDown is called when the user has pressed the mouse button

	while over the GUIElement or Collider.
OnMouseDrag	OnMouseDrag is called when the user has clicked on a GUIElement or Collider and is still holding down the mouse.
OnMouseEnter	Called when the mouse enters the GUIElement or Collider.
OnMouseExit	Called when the mouse is not any longer over the GUIElement or Collider.
OnMouseOver	Called every frame while the mouse is over the GUIElement or Collider.
OnMouseUp	OnMouseUp is called when the user has released the mouse button.
OnMouseUpAsButton	OnMouseUpAsButton is only called when the mouse is released over the same GUIElement or Collider as it was pressed.
OnNetworkInstantiate	Called on objects which have been network instantiated with Network.Instantiate.
OnParticleCollision	OnParticleCollision is called when a particle hits a Collider.
OnParticleSystemStopped	OnParticleSystemStopped is called when all particles in the system have died, and no new particles will be born. New particles cease to be created either after Stop is called, or when the duration property of a non-looping system has been exceeded.
OnParticleTrigger	OnParticleTrigger is called when any particles in a particle system meet the conditions in the trigger module.
OnPlayerConnected	Called on the server whenever a new player has successfully connected.
OnPlayerDisconnected	Called on the server whenever a player disconnected from the server.
OnPostRender	OnPostRender is called after a camera finished rendering the Scene.
OnPreCull	OnPreCull is called before a camera culls the Scene.
OnPreRender	OnPreRender is called before a camera starts rendering the Scene.
OnRenderImage	OnRenderImage is called after all rendering is complete to render image.
OnRenderObject	OnRenderObject is called after camera has rendered the Scene.
OnSerializeNetworkView	Used to customize synchronization of variables in a script watched by a network view.
OnServerInitialized	Called on the server whenever a Network.InitializeServer was invoked and has completed.
OnTransformChildrenChanged	This function is called when the list of children of the transform of the GameObject has changed.

OnTransformParentChanged	This function is called when the parent property of the transform of the GameObject has changed.
OnTriggerEnter	OnTriggerEnter is called when the Collider other enters the trigger.
OnTriggerEnter2D	Sent when another object enters a trigger collider attached to this object (2D physics only).
OnTriggerExit	OnTriggerExit is called when the Collider other has stopped touching the trigger.
OnTriggerExit2D	Sent when another object leaves a trigger collider attached to this object (2D physics only).
OnTriggerStay	OnTriggerStay is called once per physics update for every Collider other that is touching the trigger.
OnTriggerStay2D	Sent each frame where another object is within a trigger collider attached to this object (2D physics only).
OnValidate	This function is called when the script is loaded or a value is changed in the inspector (Called in the editor only).
OnWillRenderObject	OnWillRenderObject is called for each camera if the object is visible and not a UI element.
Reset	Reset to default values.
Start	Start is called on the frame when a script is enabled just before any of the Update methods are called the first time.
Update	Update is called every frame, if the MonoBehaviour is enabled.

Obsolete

Description

An extension for the NetworkManager that displays a default HUD for controlling the network state of the game.

This component also shows useful internal state for the networking system in the inspector window of the editor. It allows users to view connections, networked objects, message handlers, and packet statistics. This information can be helpful when debugging networked games.

Inherited Members

Properties

enabled	Enabled Behaviours are Updated, disabled Behaviours are not.
isActiveAndEnabled	Has the Behaviour had active and enabled called?
gameObject	The game object this component is attached to. A component is always attached to a game object.
tag	The tag of this game object.
transform	The Transform attached to this GameObject.
runInEditMode	Allow a specific instance of a MonoBehaviour to run in edit mode (only available in the editor).
useGUILayout	Disabling this lets you skip the GUI layout phase.
hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.

Public Methods

BroadcastMessage	Calls the method named methodName on every MonoBehaviour in this game object or any of its children.
CompareTag	Is this game object tagged with tag ?
GetComponent	Returns the component of Type type if the game object has one attached, null if it doesn't.
GetComponentInChildren	Returns the component of Type type in the GameObject or any of its children using depth first search.
GetComponentInParent	Returns the component of Type type in the GameObject or any of its parents.

GetComponents	Returns all components of Type type in the GameObject.
GetComponentsInChildren	Returns all components of Type type in the GameObject or any of its children.
GetComponentsInParent	Returns all components of Type type in the GameObject or any of its parents.
SendMessage	Calls the method named methodName on every MonoBehaviour in this game object.
SendMessageUpwards	Calls the method named methodName on every MonoBehaviour in this game object and on every ancestor of the behaviour.
CancelInvoke	Cancels all Invoke calls on this MonoBehaviour.
Invoke	Invokes the method methodName in time seconds.
InvokeRepeating	Invokes the method methodName in time seconds, then repeatedly every repeatRate seconds.
IsInvoking	Is any invoke on methodName pending?
StartCoroutine	Starts a coroutine.
StopAllCoroutines	Stops all coroutines running on this behaviour.
StopCoroutine	Stops the first coroutine named methodName, or the coroutine stored in routine running on this behaviour.
GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.

Static Methods

print	Logs message to the Unity Console (identical to Debug.Log).
Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Messages

Awake	Awake is called when the script instance is being loaded.
FixedUpdate	This function is called every fixed framerate frame, if the MonoBehaviour is enabled.
LateUpdate	LateUpdate is called every frame, if the Behaviour is enabled.
OnAnimatorIK	Callback for setting up animation IK (inverse kinematics).
OnAnimatorMove	Callback for processing animation movements for modifying root motion.
OnApplicationFocus	Sent to all GameObjects when the player gets or loses focus.
OnApplicationPause	Sent to all GameObjects when the application pauses.
OnApplicationQuit	Sent to all game objects before the application quits.
OnAudioFilterRead	If OnAudioFilterRead is implemented, Unity will insert a custom filter into the audio DSP chain.
OnBecameInvisible	OnBecameInvisible is called when the renderer is no longer visible by any camera.
OnBecameVisible	OnBecameVisible is called when the renderer became visible by any camera.
OnCollisionEnter	OnCollisionEnter is called when this collider/rigidbody has begun touching another rigidbody/collider.
OnCollisionEnter2D	Sent when an incoming collider makes contact with this object's collider (2D physics only).
OnCollisionExit	OnCollisionExit is called when this collider/rigidbody has stopped touching another rigidbody/collider.
OnCollisionExit2D	Sent when a collider on another object stops touching this object's collider (2D physics only).
OnCollisionStay	:ref::OnCollisionStay is called once per frame for every collider/rigidbody that is touching rigidbody/collider.
OnCollisionStay2D	Sent each frame where a collider on another object is touching this object's collider (2D physics only).

OnConnectedToServer	Called on the client when you have successfully connected to a server.
OnControllerColliderHit	OnControllerColliderHit is called when the controller hits a collider while performing a Move.
OnDestroy	Destroying the attached Behaviour will result in the game or Scene receiving OnDestroy.
OnDisable	This function is called when the behaviour becomes disabled.
OnDisconnectedFromServer	Called on the client when the connection was lost or you disconnected from the server.
OnDrawGizmos	Implement OnDrawGizmos if you want to draw gizmos that are also pickable and always drawn.
OnDrawGizmosSelected	Implement OnDrawGizmosSelected to draw a gizmo if the object is selected.
OnEnable	This function is called when the object becomes enabled and active.
OnFailedToConnect	Called on the client when a connection attempt fails for some reason.
OnFailedToConnectToMasterServer	Called on clients or servers when there is a problem connecting to the MasterServer.
OnGUI	OnGUI is called for rendering and handling GUI events.
OnJointBreak	Called when a joint attached to the same game object broke.
OnJointBreak2D	Called when a Joint2D attached to the same game object breaks.
OnMasterServerEvent	Called on clients or servers when reporting events from the MasterServer.
OnMouseDown	OnMouseDown is called when the user has pressed the mouse button while over the GUIElement or Collider.
OnMouseDrag	OnMouseDrag is called when the user has clicked on a GUIElement or Collider and is still holding down the mouse.
OnMouseEnter	Called when the mouse enters the GUIElement or Collider.
OnMouseExit	Called when the mouse is not any longer over the GUIElement or Collider.
OnMouseOver	Called every frame while the mouse is over the GUIElement or Collider.
OnMouseUp	OnMouseUp is called when the user has released the mouse button.
OnMouseUpAsButton	OnMouseUpAsButton is only called when the mouse is released over the same GUIElement or Collider as it was pressed.

OnNetworkInstantiate	Called on objects which have been network instantiated with Network.Instantiate.
OnParticleCollision	OnParticleCollision is called when a particle hits a Collider.
OnParticleSystemStopped	OnParticleSystemStopped is called when all particles in the system have died, and no new particles will be born. New particles cease to be created either after Stop is called, or when the duration property of a non-looping system has been exceeded.
OnParticleTrigger	OnParticleTrigger is called when any particles in a particle system meet the conditions in the trigger module.
OnPlayerConnected	Called on the server whenever a new player has successfully connected.
OnPlayerDisconnected	Called on the server whenever a player disconnected from the server.
OnPostRender	OnPostRender is called after a camera finished rendering the Scene.
OnPreCull	OnPreCull is called before a camera culls the Scene.
OnPreRender	OnPreRender is called before a camera starts rendering the Scene.
OnRenderImage	OnRenderImage is called after all rendering is complete to render image.
OnRenderObject	OnRenderObject is called after camera has rendered the Scene.
OnSerializeNetworkView	Used to customize synchronization of variables in a script watched by a network view.
OnServerInitialized	Called on the server whenever a Network.InitializeServer was invoked and has completed.
OnTransformChildrenChanged	This function is called when the list of children of the transform of the GameObject has changed.
OnTransformParentChanged	This function is called when the parent property of the transform of the GameObject has changed.
OnTriggerEnter	OnTriggerEnter is called when the Collider other enters the trigger.
OnTriggerEnter2D	Sent when another object enters a trigger collider attached to this object (2D physics only).
OnTriggerExit	OnTriggerExit is called when the Collider other has stopped touching the trigger.
OnTriggerExit2D	Sent when another object leaves a trigger collider attached to this object (2D physics only).
OnTriggerStay	OnTriggerStay is called once per physics update for every Collider other that is touching the trigger.

OnTriggerStay2D	Sent each frame where another object is within a trigger collider attached to this object (2D physics only).
OnValidate	This function is called when the script is loaded or a value is changed in the inspector (Called in the editor only).
OnWillRenderObject	OnWillRenderObject is called for each camera if the object is visible and not a UI element.
Reset	Reset to default values.
Start	Start is called on the frame when a script is enabled just before any of the Update methods are called the first time.
Update	Update is called every frame, if the MonoBehaviour is enabled.

Obsolete

Description

The details of a network message received by a client or server on a network connection.

Obsolete

Description

A component that manages the process of a new host taking over a game when the old host is lost. This is referred to as "host migration". The migration manager sends information about each peer in the game to all the clients, and when the host is lost because of a crash or network outage, the clients are able to choose a new host, and continue the game.

The old host is able to rejoin the new game on the new host.

The state of SyncVars and SyncLists on all objects with NetworkIdentities in the Scene is maintained during a host migration. This also applies to custom serialized data for objects.

All of the player objects in the game are disabled when the host is lost. Then, when the other clients rejoin the new game on the new host, the corresponding players for those clients are re-enabled on the host, and respawned on the other clients. No player state data is lost during a host migration.

This class provides a simple default UI for controlling the behaviour when the host is lost. The UI can be disabled with the showGUI property. There are a number of virtual functions that can be implemented to customize the behaviour of host migration.

Note that only data that is available to clients will be preserved during a host migration. If there is data that is only on the server, then it will not be available to the client that becomes the new host. This means data on the host that is not in SyncVars or SyncLists will not be available after a host migration.

The callback function OnStartServer is invoked for all networked objects when the client becomes a new host.

On the new host, the NetworkMigrationManager uses the function NetworkServer.BecomeNewHost() to construct a networked server Scene from the state in the current ClientScene.

The peers in a game with host migration enabled are identified by their connectionId on the server. When a client reconnects to the new host of a game, this connectionId is passed to the new host so that it can match this client with the client that was connected to the old host. This Id is set on the ClientScene as the "reconnectId".

The old host of the game, the one that crashed or lost its network connection, can also reconnect to the new game as a client. This client uses the special ReconnectId of ClientScene.ReconnectIdHost (which is zero).

Inherited Members

Properties

enabled	Enabled Behaviours are Updated, disabled Behaviours are not.
isActiveAndEnabled	Has the Behaviour had active and enabled called?

gameObject	The game object this component is attached to. A component is always attached to a game object.
tag	The tag of this game object.
transform	The Transform attached to this GameObject.
runInEditMode	Allow a specific instance of a MonoBehaviour to run in edit mode (only available in the editor).
useGUILayout	Disabling this lets you skip the GUI layout phase.
hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.

Public Methods

BroadcastMessage	Calls the method named methodName on every MonoBehaviour in this game object or any of its children.
CompareTag	Is this game object tagged with tag ?
GetComponent	Returns the component of Type type if the game object has one attached, null if it doesn't.
GetComponentInChildren	Returns the component of Type type in the GameObject or any of its children using depth first search.
GetComponentInParent	Returns the component of Type type in the GameObject or any of its parents.
GetComponents	Returns all components of Type type in the GameObject.
GetComponentsInChildren	Returns all components of Type type in the GameObject or any of its children.
GetComponentsInParent	Returns all components of Type type in the GameObject or any of its parents.
SendMessage	Calls the method named methodName on every MonoBehaviour in this game object.
SendMessageUpwards	Calls the method named methodName on every MonoBehaviour in this game object and on every ancestor of the behaviour.
CancelInvoke	Cancels all Invoke calls on this MonoBehaviour.
Invoke	Invokes the method methodName in time seconds.
InvokeRepeating	Invokes the method methodName in time seconds, then repeatedly every repeatRate seconds.
IsInvoking	Is any invoke on methodName pending?

StartCoroutine	Starts a coroutine.
StopAllCoroutines	Stops all coroutines running on this behaviour.
StopCoroutine	Stops the first coroutine named methodName, or the coroutine stored in routine running on this behaviour.
GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.

Static Methods

print	Logs message to the Unity Console (identical to Debug.Log).
Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Messages

Awake	Awake is called when the script instance is being loaded.
FixedUpdate	This function is called every fixed framerate frame, if the MonoBehaviour is enabled.
LateUpdate	LateUpdate is called every frame, if the Behaviour is enabled.
OnAnimatorIK	Callback for setting up animation IK (inverse kinematics).
OnAnimatorMove	Callback for processing animation movements for modifying root motion.

OnApplicationFocus	Sent to all GameObjects when the player gets or loses focus.
OnApplicationPause	Sent to all GameObjects when the application pauses.
OnApplicationQuit	Sent to all game objects before the application quits.
OnAudioFilterRead	If OnAudioFilterRead is implemented, Unity will insert a custom filter into the audio DSP chain.
OnBecameInvisible	OnBecameInvisible is called when the renderer is no longer visible by any camera.
OnBecameVisible	OnBecameVisible is called when the renderer became visible by any camera.
OnCollisionEnter	OnCollisionEnter is called when this collider/rigidbody has begun touching another rigidbody/collider.
OnCollisionEnter2D	Sent when an incoming collider makes contact with this object's collider (2D physics only).
OnCollisionExit	OnCollisionExit is called when this collider/rigidbody has stopped touching another rigidbody/collider.
OnCollisionExit2D	Sent when a collider on another object stops touching this object's collider (2D physics only).
OnCollisionStay	:ref::OnCollisionStay is called once per frame for every collider/rigidbody that is touching rigidbody/collider.
OnCollisionStay2D	Sent each frame where a collider on another object is touching this object's collider (2D physics only).
OnConnectedToServer	Called on the client when you have successfully connected to a server.
OnControllerColliderHit	OnControllerColliderHit is called when the controller hits a collider while performing a Move.
OnDestroy	Destroying the attached Behaviour will result in the game or Scene receiving OnDestroy.
OnDisable	This function is called when the behaviour becomes disabled.
OnDisconnectedFromServer	Called on the client when the connection was lost or you disconnected from the server.
OnDrawGizmos	Implement OnDrawGizmos if you want to draw gizmos that are also pickable and always drawn.
OnDrawGizmosSelected	Implement OnDrawGizmosSelected to draw a gizmo if the object is selected.
OnEnable	This function is called when the object becomes enabled and active.

OnFailedToConnect	Called on the client when a connection attempt fails for some reason.
OnFailedToConnectToMasterServer	Called on clients or servers when there is a problem connecting to the MasterServer.
OnGUI	OnGUI is called for rendering and handling GUI events.
OnJointBreak	Called when a joint attached to the same game object broke.
OnJointBreak2D	Called when a Joint2D attached to the same game object breaks.
OnMasterServerEvent	Called on clients or servers when reporting events from the MasterServer.
OnMouseDown	OnMouseDown is called when the user has pressed the mouse button while over the GUIElement or Collider.
OnMouseDrag	OnMouseDrag is called when the user has clicked on a GUIElement or Collider and is still holding down the mouse.
OnMouseEnter	Called when the mouse enters the GUIElement or Collider.
OnMouseExit	Called when the mouse is not any longer over the GUIElement or Collider.
OnMouseOver	Called every frame while the mouse is over the GUIElement or Collider.
OnMouseUp	OnMouseUp is called when the user has released the mouse button.
OnMouseUpAsButton	OnMouseUpAsButton is only called when the mouse is released over the same GUIElement or Collider as it was pressed.
OnNetworkInstantiate	Called on objects which have been network instantiated with Network.Instantiate.
OnParticleCollision	OnParticleCollision is called when a particle hits a Collider.
OnParticleSystemStopped	OnParticleSystemStopped is called when all particles in the system have died, and no new particles will be born. New particles cease to be created either after Stop is called, or when the duration property of a non-looping system has been exceeded.
OnParticleTrigger	OnParticleTrigger is called when any particles in a particle system meet the conditions in the trigger module.
OnPlayerConnected	Called on the server whenever a new player has successfully connected.
OnPlayerDisconnected	Called on the server whenever a player disconnected from the server.
OnPostRender	OnPostRender is called after a camera finished rendering the Scene.
OnPreCull	OnPreCull is called before a camera culls the Scene.

OnPreRender	OnPreRender is called before a camera starts rendering the Scene.
OnRenderImage	OnRenderImage is called after all rendering is complete to render image.
OnRenderObject	OnRenderObject is called after camera has rendered the Scene.
OnSerializeNetworkView	Used to customize synchronization of variables in a script watched by a network view.
OnServerInitialized	Called on the server whenever a Network.InitializeServer was invoked and has completed.
OnTransformChildrenChanged	This function is called when the list of children of the transform of the GameObject has changed.
OnTransformParentChanged	This function is called when the parent property of the transform of the GameObject has changed.
OnTriggerEnter	OnTriggerEnter is called when the Collider other enters the trigger.
OnTriggerEnter2D	Sent when another object enters a trigger collider attached to this object (2D physics only).
OnTriggerExit	OnTriggerExit is called when the Collider other has stopped touching the trigger.
OnTriggerExit2D	Sent when another object leaves a trigger collider attached to this object (2D physics only).
OnTriggerStay	OnTriggerStay is called once per physics update for every Collider other that is touching the trigger.
OnTriggerStay2D	Sent each frame where another object is within a trigger collider attached to this object (2D physics only).
OnValidate	This function is called when the script is loaded or a value is changed in the inspector (Called in the editor only).
OnWillRenderObject	OnWillRenderObject is called for each camera if the object is visible and not a UI element.
Reset	Reset to default values.
Start	Start is called on the frame when a script is enabled just before any of the Update methods are called the first time.
Update	Update is called every frame, if the MonoBehaviour is enabled.

Obsolete

Description

The player objects for connections to the old host.

This is used when clients reconnect to the new host.

Obsolete

Description

Information about a player object from another peer.

Obsolete

Description

An enumeration of how to handle Scene changes when the connection to the host is lost.

Obsolete

Description

Component that controls visibility of networked objects for players.

Any object with this component on it will not be visible to players more than a (configurable) distance away.

Inherited Members

Properties

enabled	Enabled Behaviours are Updated, disabled Behaviours are not.
isActiveAndEnabled	Has the Behaviour had active and enabled called?
gameObject	The game object this component is attached to. A component is always attached to a game object.
tag	The tag of this game object.
transform	The Transform attached to this GameObject.
runInEditMode	Allow a specific instance of a MonoBehaviour to run in edit mode (only available in the editor).
useGUILayout	Disabling this lets you skip the GUI layout phase.
hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.

Public Methods

BroadcastMessage	Calls the method named methodName on every MonoBehaviour in this game object or any of its children.
CompareTag	Is this game object tagged with tag ?
GetComponent	Returns the component of Type type if the game object has one attached, null if it doesn't.
GetComponentInChildren	Returns the component of Type type in the GameObject or any of its children using depth first search.
GetComponentInParent	Returns the component of Type type in the GameObject or any of its parents.
GetComponents	Returns all components of Type type in the GameObject.

GetComponentsInChildren	Returns all components of Type type in the GameObject or any of its children.
GetComponentsInParent	Returns all components of Type type in the GameObject or any of its parents.
SendMessage	Calls the method named methodName on every MonoBehaviour in this game object.
SendMessageUpwards	Calls the method named methodName on every MonoBehaviour in this game object and on every ancestor of the behaviour.
CancelInvoke	Cancels all Invoke calls on this MonoBehaviour.
Invoke	Invokes the method methodName in time seconds.
InvokeRepeating	Invokes the method methodName in time seconds, then repeatedly every repeatRate seconds.
IsInvoking	Is any invoke on methodName pending?
StartCoroutine	Starts a coroutine.
StopAllCoroutines	Stops all coroutines running on this behaviour.
StopCoroutine	Stops the first coroutine named methodName, or the coroutine stored in routine running on this behaviour.
GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.

Static Methods

print	Logs message to the Unity Console (identical to Debug.Log).
Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.

Operators

bool	Does the object exist?

operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Messages

Awake	Awake is called when the script instance is being loaded.
FixedUpdate	This function is called every fixed framerate frame, if the MonoBehaviour is enabled.
LateUpdate	LateUpdate is called every frame, if the Behaviour is enabled.
OnAnimatorIK	Callback for setting up animation IK (inverse kinematics).
OnAnimatorMove	Callback for processing animation movements for modifying root motion.
OnApplicationFocus	Sent to all GameObjects when the player gets or loses focus.
OnApplicationPause	Sent to all GameObjects when the application pauses.
OnApplicationQuit	Sent to all game objects before the application quits.
OnAudioFilterRead	If OnAudioFilterRead is implemented, Unity will insert a custom filter into the audio DSP chain.
OnBecameInvisible	OnBecameInvisible is called when the renderer is no longer visible by any camera.
OnBecameVisible	OnBecameVisible is called when the renderer became visible by any camera.
OnCollisionEnter	OnCollisionEnter is called when this collider/rigidbody has begun touching another rigidbody/collider.
OnCollisionEnter2D	Sent when an incoming collider makes contact with this object's collider (2D physics only).
OnCollisionExit	OnCollisionExit is called when this collider/rigidbody has stopped touching another rigidbody/collider.
OnCollisionExit2D	Sent when a collider on another object stops touching this object's collider (2D physics only).
OnCollisionStay	:ref::OnCollisionStay is called once per frame for every collider/rigidbody that is touching rigidbody/collider.
OnCollisionStay2D	Sent each frame where a collider on another object is touching this object's collider (2D physics only).
OnConnectedToServer	Called on the client when you have successfully connected to a server.

OnControllerColliderHit	OnControllerColliderHit is called when the controller hits a collider while performing a Move.
OnDestroy	Destroying the attached Behaviour will result in the game or Scene receiving OnDestroy.
OnDisable	This function is called when the behaviour becomes disabled.
OnDisconnectedFromServer	Called on the client when the connection was lost or you disconnected from the server.
OnDrawGizmos	Implement OnDrawGizmos if you want to draw gizmos that are also pickable and always drawn.
OnDrawGizmosSelected	Implement OnDrawGizmosSelected to draw a gizmo if the object is selected.
OnEnable	This function is called when the object becomes enabled and active.
OnFailedToConnect	Called on the client when a connection attempt fails for some reason.
OnFailedToConnectToMasterServer	Called on clients or servers when there is a problem connecting to the MasterServer.
OnGUI	OnGUI is called for rendering and handling GUI events.
OnJointBreak	Called when a joint attached to the same game object broke.
OnJointBreak2D	Called when a Joint2D attached to the same game object breaks.
OnMasterServerEvent	Called on clients or servers when reporting events from the MasterServer.
OnMouseDown	OnMouseDown is called when the user has pressed the mouse button while over the GUIElement or Collider.
OnMouseDrag	OnMouseDrag is called when the user has clicked on a GUIElement or Collider and is still holding down the mouse.
OnMouseEnter	Called when the mouse enters the GUIElement or Collider.
OnMouseExit	Called when the mouse is not any longer over the GUIElement or Collider.
OnMouseOver	Called every frame while the mouse is over the GUIElement or Collider.
OnMouseUp	OnMouseUp is called when the user has released the mouse button.
OnMouseUpAsButton	OnMouseUpAsButton is only called when the mouse is released over the same GUIElement or Collider as it was pressed.
OnNetworkInstantiate	Called on objects which have been network instantiated with

	Network.Instantiate.
OnParticleCollision	OnParticleCollision is called when a particle hits a Collider.
OnParticleSystemStopped	OnParticleSystemStopped is called when all particles in the system have died, and no new particles will be born. New particles cease to be created either after Stop is called, or when the duration property of a non-looping system has been exceeded.
OnParticleTrigger	OnParticleTrigger is called when any particles in a particle system meet the conditions in the trigger module.
OnPlayerConnected	Called on the server whenever a new player has successfully connected.
OnPlayerDisconnected	Called on the server whenever a player disconnected from the server.
OnPostRender	OnPostRender is called after a camera finished rendering the Scene.
OnPreCull	OnPreCull is called before a camera culls the Scene.
OnPreRender	OnPreRender is called before a camera starts rendering the Scene.
OnRenderImage	OnRenderImage is called after all rendering is complete to render image.
OnRenderObject	OnRenderObject is called after camera has rendered the Scene.
OnSerializeNetworkView	Used to customize synchronization of variables in a script watched by a network view.
OnServerInitialized	Called on the server whenever a Network.InitializeServer was invoked and has completed.
OnTransformChildrenChanged	This function is called when the list of children of the transform of the GameObject has changed.
OnTransformParentChanged	This function is called when the parent property of the transform of the GameObject has changed.
OnTriggerEnter	OnTriggerEnter is called when the Collider other enters the trigger.
OnTriggerEnter2D	Sent when another object enters a trigger collider attached to this object (2D physics only).
OnTriggerExit	OnTriggerExit is called when the Collider other has stopped touching the trigger.
OnTriggerExit2D	Sent when another object leaves a trigger collider attached to this object (2D physics only).
OnTriggerStay	OnTriggerStay is called once per physics update for every Collider other that is touching the trigger.

OnTriggerStay2D	Sent each frame where another object is within a trigger collider attached to this object (2D physics only).
OnValidate	This function is called when the script is loaded or a value is changed in the inspector (Called in the editor only).
OnWillRenderObject	OnWillRenderObject is called for each camera if the object is visible and not a UI element.
Reset	Reset to default values.
Start	Start is called on the frame when a script is enabled just before any of the Update methods are called the first time.
Update	Update is called every frame, if the MonoBehaviour is enabled.

Obsolete

Description

Enumeration of methods to use to check proximity.

Obsolete

Description

General purpose serializer for UNET (for reading byte arrays).

This class works with NetworkWriter and is used for serializing data for UNet commands, RPC calls, events and low level messages.

```
using UnityEngine;
using UnityEngine.Networking;


public class ExampleScript : MonoBehaviour
{
    // Writing data to a NetworkWriter and then
    // Converting this to a NetworkReader.
    void Start()
    {
        // The data you add to your writer must be prefixed with a message type.
        // This is in the form of a short.
        short myMsgType = 143;


        NetworkWriter writer = new NetworkWriter();


        // You start the message in your writer by passing in the message type.
        // This is a short meaning that it will take up 2 bytes at the start of
        // your message.
        writer.StartMessage(myMsgType);



        // You can now begin your message. In this case we will just use strings
        writer.Write("Test data 1");
        writer.Write("Test data 2");
        writer.Write("Test data 3");



        // Make sure to end your message with FinishMessage()
        writer.FinishMessage();
```

```
    // You can now access the data in your writer. ToArray() returns a copy
    // of the bytes that the writer is using and AsArray() returns the
    // internal array of bytes, not a copy.
    byte[] writerData = writer.ToArray();


    CreateNetworkReader(writerData);
}


void CreateNetworkReader(byte[] data)
{
    // We will create the NetworkReader using the data from our previous
    // NetworkWriter.
    NetworkReader networkReader = new NetworkReader(data);



    // The first two bytes in the buffer represent the size
    // of the message. This is equal to the NetworkReader.Length
    // minus the size of the prefix.
    byte[] readerMsgSizeData = networkReader.ReadBytes(2);
    short readerMsgSize = (short)((readerMsgSizeData[1] << 8) + readerMsgSiz
    Debug.Log(readerMsgSize);



    // The message type added in NetworkWriter.StartMessage
    // is to be read now. It is a short and so consists of
    // two bytes. It is the second two bytes on the buffer.
    byte[] readerMsgTypeData = networkReader.ReadBytes(2);
    short readerMsgType = (short)((readerMsgTypeData[1] << 8) + readerMsgTyp
    Debug.Log(readerMsgType);



    // If all of your data is of the same type (in this case the
    // data on our buffer is comprised of only strings) you can
    // read all the data from the buffer using a loop like so.
    while (networkReader.Position < networkReader.Length)
    {
        Debug.Log(networkReader.ReadString());
    }
```

```
    }
}
```

Obsolete

Description

This is used to identify networked objects in a Scene. These values are allocated in the editor and are persistent for the lifetime of the object in the Scene.

Obsolete

Description

The NetworkServer uses a NetworkServerSimple for basic network functionality and adds more game-like functionality.

NetworkServer handles remote connections from remote clients via a NetworkServerSimple instance, and also has a local connection for a local client.

The NetworkServer is a singleton. It has static convenience functions such as NetworkServer.SendToAll() and NetworkServer.Spawn() which automatically use the singleton instance.

The NetworkManager uses the NetworkServer, but it can be used without the NetworkManager.

The set of networked objects that have been spawned is managed by NetworkServer. Objects are spawned with NetworkServer.Spawn() which adds them to this set, and makes them be created on clients. Spawned objects are removed automatically when they are destroyed, or than they can be removed from the spawned set by calling NetworkServer.UnSpawn() - this does not destroy the object.

There are a number of internal messages used by NetworkServer, these are setup when NetworkServer.Listen() is called.

Static Methods

AddPlayerForConnection	When an AddPlayer message handler has received a request from a player, the server calls this to associate the player object with the connection.
BecomeHost	This allows a client that has been disconnected from a server, to become the host of a new version of the game.
ReplacePlayerForConnection	This replaces the player object for a connection with a different player object. The old player object is not destroyed.
SendBytesToReady	This sends an array of bytes to all ready players.
SendToAll	Send a message structure with the given type number to all connected clients.
SendToClient	Send a message to the client which owns the given connection ID.
SendToClientOfPlayer	Send a message to the client which owns the given player object instance.
SendToReady	Send a message structure with the given type number to only clients which are ready.
SendUnreliableToAll	Send given message structure as an unreliable message to all connected clients.

SendUnreliableToReady	Send given message structure as an unreliable message only to ready clients.
SendWriterToReady	Sends the contents of a NetworkWriter object to the ready players.
Spawn	Spawn the given game object on all clients which are ready.

Obsolete

Description

The NetworkServerSimple is a basic server class without the "game" related functionality that the NetworkServer class has.

This class has no Scene management, spawning, player objects, observers, or static interface like the NetworkServer class. It is simply a server that listens on a port, manages connections, and handles messages. There can be more than one instance of this class in a process.

Like the NetworkServer and NetworkClient classes, it allows the type of NetworkConnection class created for new connections to be specified with SetNetworkConnectionClass(), so custom types of network connections can be used with it.

This class can be used by overriding the virtual functions OnConnected, OnDisconnected and OnData; or by registering message handlers.

Obsolete

Description

This component is used to make a gameObject a starting position for spawning player objects in multiplayer games.

This object's transform will be automatically registered and unregistered with the NetworkManager as a starting position.

Inherited Members

Properties

enabled	Enabled Behaviours are Updated, disabled Behaviours are not.
isActiveAndEnabled	Has the Behaviour had active and enabled called?
gameObject	The game object this component is attached to. A component is always attached to a game object.
tag	The tag of this game object.
transform	The Transform attached to this GameObject.
runInEditMode	Allow a specific instance of a MonoBehaviour to run in edit mode (only available in the editor).
useGUILayout	Disabling this lets you skip the GUI layout phase.
hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.

Public Methods

BroadcastMessage	Calls the method named methodName on every MonoBehaviour in this game object or any of its children.
CompareTag	Is this game object tagged with tag ?
GetComponent	Returns the component of Type type if the game object has one attached, null if it doesn't.
GetComponentInChildren	Returns the component of Type type in the GameObject or any of its children using depth first search.
GetComponentInParent	Returns the component of Type type in the GameObject or any of its parents.

GetComponents	Returns all components of Type type in the GameObject.
GetComponentsInChildren	Returns all components of Type type in the GameObject or any of its children.
GetComponentsInParent	Returns all components of Type type in the GameObject or any of its parents.
SendMessage	Calls the method named methodName on every MonoBehaviour in this game object.
SendMessageUpwards	Calls the method named methodName on every MonoBehaviour in this game object and on every ancestor of the behaviour.
CancelInvoke	Cancels all Invoke calls on this MonoBehaviour.
Invoke	Invokes the method methodName in time seconds.
InvokeRepeating	Invokes the method methodName in time seconds, then repeatedly every repeatRate seconds.
IsInvoking	Is any invoke on methodName pending?
StartCoroutine	Starts a coroutine.
StopAllCoroutines	Stops all coroutines running on this behaviour.
StopCoroutine	Stops the first coroutine named methodName, or the coroutine stored in routine running on this behaviour.
GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.

Static Methods

print	Logs message to the Unity Console (identical to Debug.Log).
Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Messages

Awake	Awake is called when the script instance is being loaded.
FixedUpdate	This function is called every fixed framerate frame, if the MonoBehaviour is enabled.
LateUpdate	LateUpdate is called every frame, if the Behaviour is enabled.
OnAnimatorIK	Callback for setting up animation IK (inverse kinematics).
OnAnimatorMove	Callback for processing animation movements for modifying root motion.
OnApplicationFocus	Sent to all GameObjects when the player gets or loses focus.
OnApplicationPause	Sent to all GameObjects when the application pauses.
OnApplicationQuit	Sent to all game objects before the application quits.
OnAudioFilterRead	If OnAudioFilterRead is implemented, Unity will insert a custom filter into the audio DSP chain.
OnBecameInvisible	OnBecameInvisible is called when the renderer is no longer visible by any camera.
OnBecameVisible	OnBecameVisible is called when the renderer became visible by any camera.
OnCollisionEnter	OnCollisionEnter is called when this collider/rigidbody has begun touching another rigidbody/collider.
OnCollisionEnter2D	Sent when an incoming collider makes contact with this object's collider (2D physics only).
OnCollisionExit	OnCollisionExit is called when this collider/rigidbody has stopped touching another rigidbody/collider.
OnCollisionExit2D	Sent when a collider on another object stops touching this object's collider (2D physics only).
OnCollisionStay	:ref::OnCollisionStay is called once per frame for every collider/rigidbody that is touching rigidbody/collider.
OnCollisionStay2D	Sent each frame where a collider on another object is touching this object's collider (2D physics only).

OnConnectedToServer	Called on the client when you have successfully connected to a server.
OnControllerColliderHit	OnControllerColliderHit is called when the controller hits a collider while performing a Move.
OnDestroy	Destroying the attached Behaviour will result in the game or Scene receiving OnDestroy.
OnDisable	This function is called when the behaviour becomes disabled.
OnDisconnectedFromServer	Called on the client when the connection was lost or you disconnected from the server.
OnDrawGizmos	Implement OnDrawGizmos if you want to draw gizmos that are also pickable and always drawn.
OnDrawGizmosSelected	Implement OnDrawGizmosSelected to draw a gizmo if the object is selected.
OnEnable	This function is called when the object becomes enabled and active.
OnFailedToConnect	Called on the client when a connection attempt fails for some reason.
OnFailedToConnectToMasterServer	Called on clients or servers when there is a problem connecting to the MasterServer.
OnGUI	OnGUI is called for rendering and handling GUI events.
OnJointBreak	Called when a joint attached to the same game object broke.
OnJointBreak2D	Called when a Joint2D attached to the same game object breaks.
OnMasterServerEvent	Called on clients or servers when reporting events from the MasterServer.
OnMouseDown	OnMouseDown is called when the user has pressed the mouse button while over the GUIElement or Collider.
OnMouseDrag	OnMouseDrag is called when the user has clicked on a GUIElement or Collider and is still holding down the mouse.
OnMouseEnter	Called when the mouse enters the GUIElement or Collider.
OnMouseExit	Called when the mouse is not any longer over the GUIElement or Collider.
OnMouseOver	Called every frame while the mouse is over the GUIElement or Collider.
OnMouseUp	OnMouseUp is called when the user has released the mouse button.
OnMouseUpAsButton	OnMouseUpAsButton is only called when the mouse is released over the same GUIElement or Collider as it was pressed.

OnNetworkInstantiate	Called on objects which have been network instantiated with Network.Instantiate.
OnParticleCollision	OnParticleCollision is called when a particle hits a Collider.
OnParticleSystemStopped	OnParticleSystemStopped is called when all particles in the system have died, and no new particles will be born. New particles cease to be created either after Stop is called, or when the duration property of a non-looping system has been exceeded.
OnParticleTrigger	OnParticleTrigger is called when any particles in a particle system meet the conditions in the trigger module.
OnPlayerConnected	Called on the server whenever a new player has successfully connected.
OnPlayerDisconnected	Called on the server whenever a player disconnected from the server.
OnPostRender	OnPostRender is called after a camera finished rendering the Scene.
OnPreCull	OnPreCull is called before a camera culls the Scene.
OnPreRender	OnPreRender is called before a camera starts rendering the Scene.
OnRenderImage	OnRenderImage is called after all rendering is complete to render image.
OnRenderObject	OnRenderObject is called after camera has rendered the Scene.
OnSerializeNetworkView	Used to customize synchronization of variables in a script watched by a network view.
OnServerInitialized	Called on the server whenever a Network.InitializeServer was invoked and has completed.
OnTransformChildrenChanged	This function is called when the list of children of the transform of the GameObject has changed.
OnTransformParentChanged	This function is called when the parent property of the transform of the GameObject has changed.
OnTriggerEnter	OnTriggerEnter is called when the Collider other enters the trigger.
OnTriggerEnter2D	Sent when another object enters a trigger collider attached to this object (2D physics only).
OnTriggerExit	OnTriggerExit is called when the Collider other has stopped touching the trigger.
OnTriggerExit2D	Sent when another object leaves a trigger collider attached to this object (2D physics only).
OnTriggerStay	OnTriggerStay is called once per physics update for every Collider other that is touching the trigger.

OnTriggerStay2D	Sent each frame where another object is within a trigger collider attached to this object (2D physics only).
OnValidate	This function is called when the script is loaded or a value is changed in the inspector (Called in the editor only).
OnWillRenderObject	OnWillRenderObject is called for each camera if the object is visible and not a UI element.
Reset	Reset to default values.
Start	Start is called on the frame when a script is enabled just before any of the Update methods are called the first time.
Update	Update is called every frame, if the MonoBehaviour is enabled.

Obsolete

Description

A component to synchronize the position and rotation of networked objects.

The movement of game objects can be networked by this component. There are two models of authority for networked movement:

If the object has authority on the client, then it should be controlled locally on the owning client, then movement state information will be sent from the owning client to the server, then broadcast to all of the other clients. This is common for player objects.

If the object has authority on the server, then it should be controlled on the server and movement state information will be sent to all clients. This is common for objects not related to a specific client, such as an enemy unit.

Inherited Members

Properties

enabled	Enabled Behaviours are Updated, disabled Behaviours are not.
isActiveAndEnabled	Has the Behaviour had active and enabled called?
gameObject	The game object this component is attached to. A component is always attached to a game object.
tag	The tag of this game object.
transform	The Transform attached to this GameObject.
runInEditMode	Allow a specific instance of a MonoBehaviour to run in edit mode (only available in the editor).
useGUILayout	Disabling this lets you skip the GUI layout phase.
hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.

Public Methods

BroadcastMessage	Calls the method named methodName on every MonoBehaviour in this game object or any of its children.
CompareTag	Is this game object tagged with tag ?

GetComponent	Returns the component of Type type if the game object has one attached, null if it doesn't.
GetComponentInChildren	Returns the component of Type type in the GameObject or any of its children using depth first search.
GetComponentInParent	Returns the component of Type type in the GameObject or any of its parents.
GetComponents	Returns all components of Type type in the GameObject.
GetComponentsInChildren	Returns all components of Type type in the GameObject or any of its children.
GetComponentsInParent	Returns all components of Type type in the GameObject or any of its parents.
SendMessage	Calls the method named methodName on every MonoBehaviour in this game object.
SendMessageUpwards	Calls the method named methodName on every MonoBehaviour in this game object and on every ancestor of the behaviour.
CancelInvoke	Cancels all Invoke calls on this MonoBehaviour.
Invoke	Invokes the method methodName in time seconds.
InvokeRepeating	Invokes the method methodName in time seconds, then repeatedly every repeatRate seconds.
IsInvoking	Is any invoke on methodName pending?
StartCoroutine	Starts a coroutine.
StopAllCoroutines	Stops all coroutines running on this behaviour.
StopCoroutine	Stops the first coroutine named methodName, or the coroutine stored in routine running on this behaviour.
GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.

Static Methods

print	Logs message to the Unity Console (identical to Debug.Log).
Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.

FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Messages

Awake	Awake is called when the script instance is being loaded.
FixedUpdate	This function is called every fixed framerate frame, if the MonoBehaviour is enabled.
LateUpdate	LateUpdate is called every frame, if the Behaviour is enabled.
OnAnimatorIK	Callback for setting up animation IK (inverse kinematics).
OnAnimatorMove	Callback for processing animation movements for modifying root motion.
OnApplicationFocus	Sent to all GameObjects when the player gets or loses focus.
OnApplicationPause	Sent to all GameObjects when the application pauses.
OnApplicationQuit	Sent to all game objects before the application quits.
OnAudioFilterRead	If OnAudioFilterRead is implemented, Unity will insert a custom filter into the audio DSP chain.
OnBecameInvisible	OnBecameInvisible is called when the renderer is no longer visible by any camera.
OnBecameVisible	OnBecameVisible is called when the renderer became visible by any camera.
OnCollisionEnter	OnCollisionEnter is called when this collider/rigidbody has begun touching another rigidbody/collider.
OnCollisionEnter2D	Sent when an incoming collider makes contact with this object's collider (2D physics only).
OnCollisionExit	OnCollisionExit is called when this collider/rigidbody has stopped touching another rigidbody/collider.
OnCollisionExit2D	Sent when a collider on another object stops touching this object's

	collider (2D physics only).
OnCollisionStay	:ref::OnCollisionStay is called once per frame for every collider/rigidbody that is touching rigidbody/collider.
OnCollisionStay2D	Sent each frame where a collider on another object is touching this object's collider (2D physics only).
OnConnectedToServer	Called on the client when you have successfully connected to a server.
OnControllerColliderHit	OnControllerColliderHit is called when the controller hits a collider while performing a Move.
OnDestroy	Destroying the attached Behaviour will result in the game or Scene receiving OnDestroy.
OnDisable	This function is called when the behaviour becomes disabled.
OnDisconnectedFromServer	Called on the client when the connection was lost or you disconnected from the server.
OnDrawGizmos	Implement OnDrawGizmos if you want to draw gizmos that are also pickable and always drawn.
OnDrawGizmosSelected	Implement OnDrawGizmosSelected to draw a gizmo if the object is selected.
OnEnable	This function is called when the object becomes enabled and active.
OnFailedToConnect	Called on the client when a connection attempt fails for some reason.
OnFailedToConnectToMasterServer	Called on clients or servers when there is a problem connecting to the MasterServer.
OnGUI	OnGUI is called for rendering and handling GUI events.
OnJointBreak	Called when a joint attached to the same game object broke.
OnJointBreak2D	Called when a Joint2D attached to the same game object breaks.
OnMasterServerEvent	Called on clients or servers when reporting events from the MasterServer.
OnMouseDown	OnMouseDown is called when the user has pressed the mouse button while over the GUIElement or Collider.
OnMouseDrag	OnMouseDrag is called when the user has clicked on a GUIElement or Collider and is still holding down the mouse.
OnMouseEnter	Called when the mouse enters the GUIElement or Collider.
OnMouseExit	Called when the mouse is not any longer over the GUIElement or Collider.

OnMouseOver	Called every frame while the mouse is over the GUIElement or Collider.
OnMouseUp	OnMouseUp is called when the user has released the mouse button.
OnMouseUpAsButton	OnMouseUpAsButton is only called when the mouse is released over the same GUIElement or Collider as it was pressed.
OnNetworkInstantiate	Called on objects which have been network instantiated with Network.Instantiate.
OnParticleCollision	OnParticleCollision is called when a particle hits a Collider.
OnParticleSystemStopped	OnParticleSystemStopped is called when all particles in the system have died, and no new particles will be born. New particles cease to be created either after Stop is called, or when the duration property of a non-looping system has been exceeded.
OnParticleTrigger	OnParticleTrigger is called when any particles in a particle system meet the conditions in the trigger module.
OnPlayerConnected	Called on the server whenever a new player has successfully connected.
OnPlayerDisconnected	Called on the server whenever a player disconnected from the server.
OnPostRender	OnPostRender is called after a camera finished rendering the Scene.
OnPreCull	OnPreCull is called before a camera culls the Scene.
OnPreRender	OnPreRender is called before a camera starts rendering the Scene.
OnRenderImage	OnRenderImage is called after all rendering is complete to render image.
OnRenderObject	OnRenderObject is called after camera has rendered the Scene.
OnSerializeNetworkView	Used to customize synchronization of variables in a script watched by a network view.
OnServerInitialized	Called on the server whenever a Network.InitializeServer was invoked and has completed.
OnTransformChildrenChanged	This function is called when the list of children of the transform of the GameObject has changed.
OnTransformParentChanged	This function is called when the parent property of the transform of the GameObject has changed.
OnTriggerEnter	OnTriggerEnter is called when the Collider other enters the trigger.
OnTriggerEnter2D	Sent when another object enters a trigger collider attached to this object (2D physics only).
OnTriggerExit	OnTriggerExit is called when the Collider other has stopped touching

	the trigger.
OnTriggerExit2D	Sent when another object leaves a trigger collider attached to this object (2D physics only).
OnTriggerStay	OnTriggerStay is called once per physics update for every Collider other that is touching the trigger.
OnTriggerStay2D	Sent each frame where another object is within a trigger collider attached to this object (2D physics only).
OnValidate	This function is called when the script is loaded or a value is changed in the inspector (Called in the editor only).
OnWillRenderObject	OnWillRenderObject is called for each camera if the object is visible and not a UI element.
Reset	Reset to default values.
Start	Start is called on the frame when a script is enabled just before any of the Update methods are called the first time.
Update	Update is called every frame, if the MonoBehaviour is enabled.

Obsolete

Description

An axis or set of axis.

Obsolete

Description

How much to compress sync data.

Obsolete

Description

How to synchronize an object's position.

Obsolete

Description

A component to synchronize the position of child transforms of networked objects.

There must be a NetworkTransform on the root object of the hierarchy. There can be multiple NetworkTransformChild components on an object. This does not use physics for synchronization, it simply synchronizes the localPosition and localRotation of the child transform and lerps towards the recieved values.

Inherited Members

Properties

enabled	Enabled Behaviours are Updated, disabled Behaviours are not.
isActiveAndEnabled	Has the Behaviour had active and enabled called?
gameObject	The game object this component is attached to. A component is always attached to a game object.
tag	The tag of this game object.
transform	The Transform attached to this GameObject.
runInEditMode	Allow a specific instance of a MonoBehaviour to run in edit mode (only available in the editor).
useGUILayout	Disabling this lets you skip the GUI layout phase.
hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.

Public Methods

BroadcastMessage	Calls the method named methodName on every MonoBehaviour in this game object or any of its children.
CompareTag	Is this game object tagged with tag ?
GetComponent	Returns the component of Type type if the game object has one attached, null if it doesn't.
GetComponentInChildren	Returns the component of Type type in the GameObject or any of its children using depth first search.
GetComponentInParent	Returns the component of Type type in the GameObject or any of its parents.

GetComponents	Returns all components of Type type in the GameObject.
GetComponentsInChildren	Returns all components of Type type in the GameObject or any of its children.
GetComponentsInParent	Returns all components of Type type in the GameObject or any of its parents.
SendMessage	Calls the method named methodName on every MonoBehaviour in this game object.
SendMessageUpwards	Calls the method named methodName on every MonoBehaviour in this game object and on every ancestor of the behaviour.
CancelInvoke	Cancels all Invoke calls on this MonoBehaviour.
Invoke	Invokes the method methodName in time seconds.
InvokeRepeating	Invokes the method methodName in time seconds, then repeatedly every repeatRate seconds.
IsInvoking	Is any invoke on methodName pending?
StartCoroutine	Starts a coroutine.
StopAllCoroutines	Stops all coroutines running on this behaviour.
StopCoroutine	Stops the first coroutine named methodName, or the coroutine stored in routine running on this behaviour.
GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.

Static Methods

print	Logs message to the Unity Console (identical to Debug.Log).
Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Messages

Awake	Awake is called when the script instance is being loaded.
FixedUpdate	This function is called every fixed framerate frame, if the MonoBehaviour is enabled.
LateUpdate	LateUpdate is called every frame, if the Behaviour is enabled.
OnAnimatorIK	Callback for setting up animation IK (inverse kinematics).
OnAnimatorMove	Callback for processing animation movements for modifying root motion.
OnApplicationFocus	Sent to all GameObjects when the player gets or loses focus.
OnApplicationPause	Sent to all GameObjects when the application pauses.
OnApplicationQuit	Sent to all game objects before the application quits.
OnAudioFilterRead	If OnAudioFilterRead is implemented, Unity will insert a custom filter into the audio DSP chain.
OnBecameInvisible	OnBecameInvisible is called when the renderer is no longer visible by any camera.
OnBecameVisible	OnBecameVisible is called when the renderer became visible by any camera.
OnCollisionEnter	OnCollisionEnter is called when this collider/rigidbody has begun touching another rigidbody/collider.
OnCollisionEnter2D	Sent when an incoming collider makes contact with this object's collider (2D physics only).
OnCollisionExit	OnCollisionExit is called when this collider/rigidbody has stopped touching another rigidbody/collider.
OnCollisionExit2D	Sent when a collider on another object stops touching this object's collider (2D physics only).
OnCollisionStay	:ref::OnCollisionStay is called once per frame for every collider/rigidbody that is touching rigidbody/collider.
OnCollisionStay2D	Sent each frame where a collider on another object is touching this object's collider (2D physics only).

OnConnectedToServer	Called on the client when you have successfully connected to a server.
OnControllerColliderHit	OnControllerColliderHit is called when the controller hits a collider while performing a Move.
OnDestroy	Destroying the attached Behaviour will result in the game or Scene receiving OnDestroy.
OnDisable	This function is called when the behaviour becomes disabled.
OnDisconnectedFromServer	Called on the client when the connection was lost or you disconnected from the server.
OnDrawGizmos	Implement OnDrawGizmos if you want to draw gizmos that are also pickable and always drawn.
OnDrawGizmosSelected	Implement OnDrawGizmosSelected to draw a gizmo if the object is selected.
OnEnable	This function is called when the object becomes enabled and active.
OnFailedToConnect	Called on the client when a connection attempt fails for some reason.
OnFailedToConnectToMasterServer	Called on clients or servers when there is a problem connecting to the MasterServer.
OnGUI	OnGUI is called for rendering and handling GUI events.
OnJointBreak	Called when a joint attached to the same game object broke.
OnJointBreak2D	Called when a Joint2D attached to the same game object breaks.
OnMasterServerEvent	Called on clients or servers when reporting events from the MasterServer.
OnMouseDown	OnMouseDown is called when the user has pressed the mouse button while over the GUIElement or Collider.
OnMouseDrag	OnMouseDrag is called when the user has clicked on a GUIElement or Collider and is still holding down the mouse.
OnMouseEnter	Called when the mouse enters the GUIElement or Collider.
OnMouseExit	Called when the mouse is not any longer over the GUIElement or Collider.
OnMouseOver	Called every frame while the mouse is over the GUIElement or Collider.
OnMouseUp	OnMouseUp is called when the user has released the mouse button.
OnMouseUpAsButton	OnMouseUpAsButton is only called when the mouse is released over the same GUIElement or Collider as it was pressed.

OnNetworkInstantiate	Called on objects which have been network instantiated with Network.Instantiate.
OnParticleCollision	OnParticleCollision is called when a particle hits a Collider.
OnParticleSystemStopped	OnParticleSystemStopped is called when all particles in the system have died, and no new particles will be born. New particles cease to be created either after Stop is called, or when the duration property of a non-looping system has been exceeded.
OnParticleTrigger	OnParticleTrigger is called when any particles in a particle system meet the conditions in the trigger module.
OnPlayerConnected	Called on the server whenever a new player has successfully connected.
OnPlayerDisconnected	Called on the server whenever a player disconnected from the server.
OnPostRender	OnPostRender is called after a camera finished rendering the Scene.
OnPreCull	OnPreCull is called before a camera culls the Scene.
OnPreRender	OnPreRender is called before a camera starts rendering the Scene.
OnRenderImage	OnRenderImage is called after all rendering is complete to render image.
OnRenderObject	OnRenderObject is called after camera has rendered the Scene.
OnSerializeNetworkView	Used to customize synchronization of variables in a script watched by a network view.
OnServerInitialized	Called on the server whenever a Network.InitializeServer was invoked and has completed.
OnTransformChildrenChanged	This function is called when the list of children of the transform of the GameObject has changed.
OnTransformParentChanged	This function is called when the parent property of the transform of the GameObject has changed.
OnTriggerEnter	OnTriggerEnter is called when the Collider other enters the trigger.
OnTriggerEnter2D	Sent when another object enters a trigger collider attached to this object (2D physics only).
OnTriggerExit	OnTriggerExit is called when the Collider other has stopped touching the trigger.
OnTriggerExit2D	Sent when another object leaves a trigger collider attached to this object (2D physics only).
OnTriggerStay	OnTriggerStay is called once per physics update for every Collider other that is touching the trigger.

OnTriggerStay2D	Sent each frame where another object is within a trigger collider attached to this object (2D physics only).
OnValidate	This function is called when the script is loaded or a value is changed in the inspector (Called in the editor only).
OnWillRenderObject	OnWillRenderObject is called for each camera if the object is visible and not a UI element.
Reset	Reset to default values.
Start	Start is called on the frame when a script is enabled just before any of the Update methods are called the first time.
Update	Update is called every frame, if the MonoBehaviour is enabled.

Obsolete

Description

This is a helper component to help understand and debug networked movement synchronization with the NetworkTransform component.

Inherited Members

Properties

enabled	Enabled Behaviours are Updated, disabled Behaviours are not.
isActiveAndEnabled	Has the Behaviour had active and enabled called?
gameObject	The game object this component is attached to. A component is always attached to a game object.
tag	The tag of this game object.
transform	The Transform attached to this GameObject.
runInEditMode	Allow a specific instance of a MonoBehaviour to run in edit mode (only available in the editor).
useGUILayout	Disabling this lets you skip the GUI layout phase.
hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.

Public Methods

BroadcastMessage	Calls the method named methodName on every MonoBehaviour in this game object or any of its children.
CompareTag	Is this game object tagged with tag ?
GetComponent	Returns the component of Type type if the game object has one attached, null if it doesn't.
GetComponentInChildren	Returns the component of Type type in the GameObject or any of its children using depth first search.
GetComponentInParent	Returns the component of Type type in the GameObject or any of its parents.
GetComponents	Returns all components of Type type in the GameObject.
GetComponentsInChildren	Returns all components of Type type in the GameObject or any of its children.

GetComponentsInParent	Returns all components of Type type in the GameObject or any of its parents.
SendMessage	Calls the method named methodName on every MonoBehaviour in this game object.
SendMessageUpwards	Calls the method named methodName on every MonoBehaviour in this game object and on every ancestor of the behaviour.
CancelInvoke	Cancels all Invoke calls on this MonoBehaviour.
Invoke	Invokes the method methodName in time seconds.
InvokeRepeating	Invokes the method methodName in time seconds, then repeatedly every repeatRate seconds.
IsInvoking	Is any invoke on methodName pending?
StartCoroutine	Starts a coroutine.
StopAllCoroutines	Stops all coroutines running on this behaviour.
StopCoroutine	Stops the first coroutine named methodName, or the coroutine stored in routine running on this behaviour.
GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.

Static Methods

print	Logs message to the Unity Console (identical to Debug.Log).
Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.

operator ==	Compares two object references to see if they refer to the same object.

Messages

Awake	Awake is called when the script instance is being loaded.
FixedUpdate	This function is called every fixed framerate frame, if the MonoBehaviour is enabled.
LateUpdate	LateUpdate is called every frame, if the Behaviour is enabled.
OnAnimatorIK	Callback for setting up animation IK (inverse kinematics).
OnAnimatorMove	Callback for processing animation movements for modifying root motion.
OnApplicationFocus	Sent to all GameObjects when the player gets or loses focus.
OnApplicationPause	Sent to all GameObjects when the application pauses.
OnApplicationQuit	Sent to all game objects before the application quits.
OnAudioFilterRead	If OnAudioFilterRead is implemented, Unity will insert a custom filter into the audio DSP chain.
OnBecameInvisible	OnBecameInvisible is called when the renderer is no longer visible by any camera.
OnBecameVisible	OnBecameVisible is called when the renderer became visible by any camera.
OnCollisionEnter	OnCollisionEnter is called when this collider/rigidbody has begun touching another rigidbody/collider.
OnCollisionEnter2D	Sent when an incoming collider makes contact with this object's collider (2D physics only).
OnCollisionExit	OnCollisionExit is called when this collider/rigidbody has stopped touching another rigidbody/collider.
OnCollisionExit2D	Sent when a collider on another object stops touching this object's collider (2D physics only).
OnCollisionStay	:ref::OnCollisionStay is called once per frame for every collider/rigidbody that is touching rigidbody/collider.
OnCollisionStay2D	Sent each frame where a collider on another object is touching this object's collider (2D physics only).
OnConnectedToServer	Called on the client when you have successfully connected to a server.
OnControllerColliderHit	OnControllerColliderHit is called when the controller hits a collider while

	performing a Move.
OnDestroy	Destroying the attached Behaviour will result in the game or Scene receiving OnDestroy.
OnDisable	This function is called when the behaviour becomes disabled.
OnDisconnectedFromServer	Called on the client when the connection was lost or you disconnected from the server.
OnDrawGizmos	Implement OnDrawGizmos if you want to draw gizmos that are also pickable and always drawn.
OnDrawGizmosSelected	Implement OnDrawGizmosSelected to draw a gizmo if the object is selected.
OnEnable	This function is called when the object becomes enabled and active.
OnFailedToConnect	Called on the client when a connection attempt fails for some reason.
OnFailedToConnectToMasterServer	Called on clients or servers when there is a problem connecting to the MasterServer.
OnGUI	OnGUI is called for rendering and handling GUI events.
OnJointBreak	Called when a joint attached to the same game object broke.
OnJointBreak2D	Called when a Joint2D attached to the same game object breaks.
OnMasterServerEvent	Called on clients or servers when reporting events from the MasterServer.
OnMouseDown	OnMouseDown is called when the user has pressed the mouse button while over the GUIElement or Collider.
OnMouseDrag	OnMouseDrag is called when the user has clicked on a GUIElement or Collider and is still holding down the mouse.
OnMouseEnter	Called when the mouse enters the GUIElement or Collider.
OnMouseExit	Called when the mouse is not any longer over the GUIElement or Collider.
OnMouseOver	Called every frame while the mouse is over the GUIElement or Collider.
OnMouseUp	OnMouseUp is called when the user has released the mouse button.
OnMouseUpAsButton	OnMouseUpAsButton is only called when the mouse is released over the same GUIElement or Collider as it was pressed.
OnNetworkInstantiate	Called on objects which have been network instantiated with Network.Instantiate.

OnParticleCollision	OnParticleCollision is called when a particle hits a Collider.
OnParticleSystemStopped	OnParticleSystemStopped is called when all particles in the system have died, and no new particles will be born. New particles cease to be created either after Stop is called, or when the duration property of a non-looping system has been exceeded.
OnParticleTrigger	OnParticleTrigger is called when any particles in a particle system meet the conditions in the trigger module.
OnPlayerConnected	Called on the server whenever a new player has successfully connected.
OnPlayerDisconnected	Called on the server whenever a player disconnected from the server.
OnPostRender	OnPostRender is called after a camera finished rendering the Scene.
OnPreCull	OnPreCull is called before a camera culls the Scene.
OnPreRender	OnPreRender is called before a camera starts rendering the Scene.
OnRenderImage	OnRenderImage is called after all rendering is complete to render image.
OnRenderObject	OnRenderObject is called after camera has rendered the Scene.
OnSerializeNetworkView	Used to customize synchronization of variables in a script watched by a network view.
OnServerInitialized	Called on the server whenever a Network.InitializeServer was invoked and has completed.
OnTransformChildrenChanged	This function is called when the list of children of the transform of the GameObject has changed.
OnTransformParentChanged	This function is called when the parent property of the transform of the GameObject has changed.
OnTriggerEnter	OnTriggerEnter is called when the Collider other enters the trigger.
OnTriggerEnter2D	Sent when another object enters a trigger collider attached to this object (2D physics only).
OnTriggerExit	OnTriggerExit is called when the Collider other has stopped touching the trigger.
OnTriggerExit2D	Sent when another object leaves a trigger collider attached to this object (2D physics only).
OnTriggerStay	OnTriggerStay is called once per physics update for every Collider other that is touching the trigger.
OnTriggerStay2D	Sent each frame where another object is within a trigger collider attached to this object (2D physics only).

OnValidate	This function is called when the script is loaded or a value is changed in the inspector (Called in the editor only).
OnWillRenderObject	OnWillRenderObject is called for each camera if the object is visible and not a UI element.
Reset	Reset to default values.
Start	Start is called on the frame when a script is enabled just before any of the Update methods are called the first time.
Update	Update is called every frame, if the MonoBehaviour is enabled.

Obsolete

Description

Transport Layer API.

Obsolete

Description

General purpose serializer for UNET (for serializing data to byte arrays).

```
using UnityEngine;
using UnityEngine.Networking;


public class ExampleScript : MonoBehaviour
{
    // Writing data to a NetworkWriter and then
    // Converting this to a NetworkReader.
    void Start()
    {
        // The data you add to your writer must be prefixed with a message type.
        // This is in the form of a short.
        short myMsgType = 143;


        NetworkWriter writer = new NetworkWriter();


        // You start the message in your writer by passing in the message type.
        // This is a short meaning that it will take up 2 bytes at the start of
        // your message.
        writer.StartMessage(myMsgType);



        // You can now begin your message. In this case we will just use strings
        writer.Write("Test data 1");
        writer.Write("Test data 2");
        writer.Write("Test data 3");



        // Make sure to end your message with FinishMessage()
        writer.FinishMessage();



        // You can now access the data in your writer. ToArray() returns a copy
```

```
        // of the bytes that the writer is using and AsArray() returns the
        // internal array of bytes, not a copy.
        byte[] writerData = writer.ToArray();


        CreateNetworkReader(writerData);
    }


    void CreateNetworkReader(byte[] data)
    {
        // We will create the NetworkReader using the data from our previous
        // NetworkWriter.
        NetworkReader networkReader = new NetworkReader(data);



        // The first two bytes in the buffer represent the size
        // of the message. This is equal to the NetworkReader.Length
        // minus the size of the prefix.
        byte[] readerMsgSizeData = networkReader.ReadBytes(2);
        short readerMsgSize = (short)((readerMsgSizeData[1] << 8) + readerMsgSiz
        Debug.Log(readerMsgSize);



        // The message type added in NetworkWriter.StartMessage
        // is to be read now. It is a short and so consists of
        // two bytes. It is the second two bytes on the buffer.
        byte[] readerMsgTypeData = networkReader.ReadBytes(2);
        short readerMsgType = (short)((readerMsgTypeData[1] << 8) + readerMsgTyp
        Debug.Log(readerMsgType);



        // If all of your data is of the same type (in this case the
        // data on our buffer is comprised of only strings) you can
        // read all the data from the buffer using a loop like so.
        while (networkReader.Position < networkReader.Length)
        {
            Debug.Log(networkReader.ReadString());
        }
    }
}
```

Obsolete

Description

This represents a networked player.

Obsolete

Description

This is the base class for type-specific SyncList classes.

A SyncList can only be of the following type;

- Basic type (byte, int, float, string, UInt64, etc)
- Built-in Unity math type (Vector3, Quaternion, etc),
- NetworkIdentity
- NetworkInstanceId
- NetworkHash128
- GameObject with a NetworkIdentity component attached.

Public Methods

HandleMsg	Internal function used for remote list operations.

Obsolete

Description

The types of operations that can occur for SyncLists.

Obsolete

Description

A list of booleans that will be synchronized from server to clients.

Inherited Members

Public Methods

HandleMsg	Internal function used for remote list operations.

Obsolete

Description

A list of floats that will be synchronized from server to clients.

Inherited Members

Public Methods

HandleMsg	Internal function used for remote list operations.

Obsolete

Description

A list of integers that will be synchronized from server to clients.

Inherited Members

Public Methods

HandleMsg	Internal function used for remote list operations.

Obsolete

Description

This is a list of strings that will be synchronized from the server to clients.

Inherited Members

Public Methods

HandleMsg	Internal function used for remote list operations.

Obsolete

Description

This class is used for lists of structs that are synchronized from the server to clients.

To use SyncListStruct, derive a new class with your struct as the generic parameter.

```
using UnityEngine;
using UnityEngine.Networking;


public class MyScript : NetworkBehaviour
{
    public struct POW
    {
        public int pow;
        public float f;
    }


    public class SyncListPOW : SyncListStruct<POW>
    {
    }


    public SyncListPOW m_pows = new SyncListPOW();
}
```

Make sure your struct members are non-static public members.
We allow the following types to be used in your struct;

- Basic type (byte, int, float, string, UInt64, etc)
- Built-in Unity math type (Vector3, Quaternion, etc),
- Arrays of basic types
- Other structs containing allowable types
- NetworkIdentity
- NetworkInstanceId
- NetworkHash128
- GameObject with a NetworkIdentity component attached.

Serialization methods will be auto-generated for this new class.

Inherited Members

Public Methods

HandleMsg	Internal function used for remote list operations.

Obsolete

Description

A list of unsigned integers that will be synchronized from server to clients.

Inherited Members

Public Methods

HandleMsg	Internal function used for remote list operations.

Description

The UnityWebRequest object is used to communicate with web servers.

UnityWebRequests handle the flow of HTTP communication with web servers. Other objects - specifically DownloadHandler and UploadHandler - govern downloading and uploading data, respectively.

A set of static functions are provided for convenience; these return UnityWebRequest objects properly configured for many common use cases. *See:* Get, Post, Put, GetTexture.

Note: Most properties on the UnityWebRequest object cannot be altered once the UnityWebRequest has begun communicating with a remote server by calling the Send method.

Static Properties

kHttpVerbCREATE	The string "CREATE", commonly used as the verb for an HTTP CREATE request.
kHttpVerbDELETE	The string "DELETE", commonly used as the verb for an HTTP DELETE request.
kHttpVerbGET	The string "GET", commonly used as the verb for an HTTP GET request.
kHttpVerbHEAD	The string "HEAD", commonly used as the verb for an HTTP HEAD request.
kHttpVerbPOST	The string "POST", commonly used as the verb for an HTTP POST request.
kHttpVerbPUT	The string "PUT", commonly used as the verb for an HTTP PUT request.

Properties

certificateHandler	Holds a reference to a CertificateHandler object, which manages certificate validation for this UnityWebRequest.
chunkedTransfer	Indicates whether the UnityWebRequest system should employ the HTTP/1.1 chunked-transfer encoding method.
disposeCertificateHandlerOnDispose	If true, any CertificateHandler attached to this UnityWebRequest will have CertificateHandler.Dispose called automatically when UnityWebRequest.Dispose is called.
disposeDownloadHandlerOnDispose	If true, any DownloadHandler attached to this UnityWebRequest will have DownloadHandler.Dispose called automatically when UnityWebRequest.Dispose is called.
disposeUploadHandlerOnDispose	If true, any UploadHandler attached to this UnityWebRequest will have UploadHandler.Dispose called automatically when UnityWebRequest.Dispose is called.

downloadedBytes	Returns the number of bytes of body data the system has downloaded from the remote server. (Read Only)
downloadHandler	Holds a reference to a DownloadHandler object, which manages body data received from the remote server by this UnityWebRequest.
downloadProgress	Returns a floating-point value between 0.0 and 1.0, indicating the progress of downloading body data from the server. (Read Only)
error	A human-readable string describing any system errors encountered by this UnityWebRequest object while handling HTTP requests or responses. (Read Only)
isDone	Returns true after the UnityWebRequest has finished communicating with the remote server. (Read Only)
isHttpError	Returns true after this UnityWebRequest receives an HTTP response code indicating an error. (Read Only)
isModifiable	Returns true while a UnityWebRequest's configuration properties can be altered. (Read Only)
isNetworkError	Returns true after this UnityWebRequest encounters a system error. (Read Only)
method	Defines the HTTP verb used by this UnityWebRequest, such as GET or POST.
redirectLimit	Indicates the number of redirects which this UnityWebRequest will follow before halting with a "Redirect Limit Exceeded" system error.
responseCode	The numeric HTTP response code returned by the server, such as 200, 404 or 500. (Read Only)
timeout	Sets UnityWebRequest to attempt to abort after the number of seconds in timeout have passed.
uploadedBytes	Returns the number of bytes of body data the system has uploaded to the remote server. (Read Only)
uploadHandler	Holds a reference to the UploadHandler object which manages body data to be uploaded to the remote server.
uploadProgress	Returns a floating-point value between 0.0 and 1.0, indicating the progress of uploading body data to the server.
uri	Defines the target URI for the UnityWebRequest to communicate with.
url	Defines the target URL for the UnityWebRequest to communicate with.
useHttpContinue	Determines whether this UnityWebRequest will include Expect: 100-Continue in its outgoing request headers. (Default: true).

Constructors

UnityWebRequest	Creates a UnityWebRequest with the default options and no attached DownloadHandler or UploadHandler. Default method is GET.

Public Methods

Abort	If in progress, halts the UnityWebRequest as soon as possible.
Dispose	Signals that this UnityWebRequest is no longer being used, and should clean up any resources it is using.
GetRequestHeader	Retrieves the value of a custom request header.
GetResponseHeader	Retrieves the value of a response header from the latest HTTP response received.
GetResponseHeaders	Retrieves a dictionary containing all the response headers received by this UnityWebRequest in the latest HTTP response.
SendWebRequest	Begin communicating with the remote server.
SetRequestHeader	Set a HTTP request header to a custom value.

Static Methods

ClearCookieCache	Clears stored cookies from the cache.
Delete	Creates a UnityWebRequest configured for HTTP DELETE.
EscapeURL	Escapes characters in a string to ensure they are URL-friendly.
GenerateBoundary	Generate a random 40-byte array for use as a multipart form boundary.
Get	Creates a UnityWebRequest configured for HTTP GET.
Head	Creates a UnityWebRequest configured to send a HTTP HEAD request.
Post	Creates a UnityWebRequest configured to send form data to a server via HTTP POST.
Put	Creates a UnityWebRequest configured to upload raw data to a remote server via HTTP PUT.
SerializeFormSections	Converts a List of IMultipartFormSection objects into a byte array containing raw multipart form data.
SerializeSimpleForm	Serialize a dictionary of strings into a byte array containing URL-encoded UTF8 characters.

UnEscapeURL	Converts URL-friendly escape sequences back to normal text.

Description

Helpers for downloading asset bundles using UnityWebRequest.

Static Methods

GetAssetBundle	Creates a UnityWebRequest optimized for downloading a Unity Asset Bundle via HTTP GET.

Description

Asynchronous operation object returned from UnityWebRequest.SendWebRequest().

You can yield until it continues, register an event handler with AsyncOperation.completed, or manually check whether it's done (AsyncOperation.isDone) or progress (AsyncOperation.progress).

Properties

webRequest	Returns the associated UnityWebRequest that created the operation.

Inherited Members

Properties

allowSceneActivation	Allow Scenes to be activated as soon as it is ready.
isDone	Has the operation finished? (Read Only)
priority	Priority lets you tweak in which order async operation calls will be performed.
progress	What's the operation's progress. (Read Only)

Events

completed	Event that is invoked upon operation completion. An event handler that is registered in the same frame as the call that creates it will be invoked next frame, even if the operation is able to complete synchronously. If a handler is registered after the operation has completed and has already invoked the complete event, the handler will be called synchronously.

Description

Helpers for downloading multimedia files using UnityWebRequest.

Static Methods

GetAudioClip	Create a UnityWebRequest to download an audio clip via HTTP GET and create an AudioClip based on the retrieved data.

Description

Helpers for downloading image files into Textures using UnityWebRequest.

Static Methods

GetTexture	Create a UnityWebRequest intended to download an image via HTTP GET and create a Texture based on the retrieved data.

Description

Helper object for UnityWebRequests. Manages the buffering and transmission of body data during HTTP requests.

When attached to a UnityWebRequest, an UploadHandler object handles all information regarding the buffering and transmission of body data during an HTTP request. By placing data in an UploadHandler and attaching it to a UnityWebRequest, the UnityWebRequest is implicitly instructed to transmit the UploadHandler's data to the remote server. The data will always be delivered as HTTP request body data.

UploadHandler is a base class and cannot be directly instantiated. Currently, only one type of UploadHandler is available: UploadHandlerRaw.

See Also: UnityWebRequest, UploadHandlerRaw.

Properties

contentType	Determines the default Content-Type header which will be transmitted with the outbound HTTP request.
data	The raw data which will be transmitted to the remote server as body data. (Read Only)
progress	Returns the proportion of data uploaded to the remote server compared to the total amount of data to upload. (Read Only)

Public Methods

Dispose	Signals that this UploadHandler is no longer being used, and should clean up any resources it is using.

Description

A specialized UploadHandler that reads data from a given file and sends raw bytes to the server as the request body.

You can use it to send a large amount of data to the server with a low memory footprint.

```
using System.Collections;
using UnityEngine;
using UnityEngine.Networking;


public class UHFileSample : MonoBehaviour
{
    void Start()
    {
        StartCoroutine(UploadFileData());
    }


    IEnumerator UploadFileData()
    {
        using (var uwr = new UnityWebRequest("http://yourwebsite.com/upload", Un
        {
            uwr.uploadHandler = new UploadHandlerFile("/path/to/file");
            yield return uwr.SendWebRequest();
            if (uwr.isNetworkError || uwr.isHttpError)
                Debug.LogError(uwr.error);
            else
            {
                // file data successfully sent
            }
        }
    }
}
```

Constructors

UploadHandlerFile	Create a new upload handler to send data from the given file to the server.

Inherited Members

Properties

contentType	Determines the default Content-Type header which will be transmitted with the outbound HTTP request.
data	The raw data which will be transmitted to the remote server as body data. (Read Only)
progress	Returns the proportion of data uploaded to the remote server compared to the total amount of data to upload. (Read Only)

Public Methods

Dispose	Signals that this UploadHandler is no longer being used, and should clean up any resources it is using.

Description

A general-purpose UploadHandler subclass, using a native-code memory buffer.

This subclass copies input data into a native-code memory buffer at construction time, and transmits that data verbatim as HTTP request body data.

Constructors

UploadHandlerRaw	General constructor. Contents of the input argument are copied into a native buffer.

Inherited Members

Properties

contentType	Determines the default Content-Type header which will be transmitted with the outbound HTTP request.
data	The raw data which will be transmitted to the remote server as body data. (Read Only)
progress	Returns the proportion of data uploaded to the remote server compared to the total amount of data to upload. (Read Only)

Public Methods

Dispose	Signals that this UploadHandler is no longer being used, and should clean up any resources it is using.

Description

Networking Utility.

Static Methods

GetAccessTokenForNetwork	Utility function to get this client's access token for a particular network, if it has been set.
GetSourceID	Utility function to get the client's SourceID for unique identification.
SetAccessTokenForNetwork	Utility function that accepts the access token for a network after it's received from the server.

Description

An interface for composition of data into multipart forms.

In order to provide a finer level of control for those wishing to generate multipart form data, but without forcing most users to refer to RFC 2388, Unity provides this simple interface which the UnityWebRequest API can use to serialize complex data into properly-formatted bytes.

For convenience, the two general types of form sections have been encapsulated into two stock implementations of IMultipartFormSection. Both stock implementations are simply controlled via their constructors.

IMultipartFormSection implementors are converted into bytes via [UnityWebRequest.SerializeFormSections].

See Also: [MultipartFormDataSection], [MultipartFormFileSection].

Properties

contentType	Returns the value to use in the Content-Type header for this form section.
fileName	Returns a string denoting the desired filename of this section on the destination server.
sectionData	Returns the raw binary data contained in this section. Must not return null or a zero-length array.
sectionName	Returns the name of this section, if any.

Description

An enumeration of the options that can be set on a network channel.

Properties

MaxPendingBuffers	The option to set the number of pending buffers for a channel.
AllowFragmentation	The option to allow packet fragmentation for a channel.
MaxPacketSize	The option to set the maximum packet size allowed for a channel.

Description

Defines size of the buffer holding reliable messages, before they will be acknowledged.

Properties

Acks32	Ack buffer can hold 32 messages.
Acks64	Ack buffer can hold 64 messages.
Acks96	Ack buffer can hold 96 messages.
Acks128	Ack buffer can hold 128 messages.

Description

Possible Networking.NetworkTransport errors.

A group of possible NetworkTransport errors. If an error is returned from a NetworkTransport function, you can use NetworkError to find out what type of error occured. Make sure to use NetworkTransport.Receive or similar to listen for these type of events.

```
//Create a Button (Create>UI>Button) for the host.


//Attach this Button in the Inspector of your GameObject.


//In Play Mode, click the Button to connect. If the connection works, the detail


using UnityEngine;
using UnityEngine.Networking;
using UnityEngine.UI;


public class Example : MonoBehaviour
{
    int m_ServerSocket;


    HostTopology m_HostTopology;


    //These are the Buttons that start the client and server, and the Button for
    //Assure that you assign these in the Inspector before testing
    public Button m_ServerButton;


    void Start()
    {
        //Set up the Connection Configuration which holds channel information
        ConnectionConfig config = new ConnectionConfig();


        //Create a new Host information based on the configuration created, and
        m_HostTopology = new HostTopology(config, 20);
```

```
        //Initialise the NetworkTransport
        NetworkTransport.Init();


        //Call the ServerButton function when you click the server Button
        m_ServerButton.onClick.AddListener(ServerButton);
    }


    void Update()
    {
        //These are the variables that are replaced by the incoming message
        int outHostId;
        int outConnectionId;
        int outChannelId;
        byte[] buffer = new byte[1024];
        int receivedSize;
        byte error;


        //Set up the Network Transport to receive the incoming message, and deci
        NetworkEventType eventType = NetworkTransport.Receive(out outHostId, out


        switch (eventType)
        {
            //Use this case when there is a connection detected
            case NetworkEventType.ConnectEvent:
            {
                //Call the function to deal with the received information
                OnConnect(outHostId, outConnectionId, (NetworkError)error);
                break;
            }


            case NetworkEventType.Nothing:
                break;


            default:
                //Output the error
                Debug.LogError("Unknown network message type received: " + event
                break;
        }
    }
```

```
    //This function is called when a connection is detected
    void OnConnect(int hostID, int connectionID, NetworkError error)
    {
        //Output the given information to the console
        Debug.Log("OnConnect(hostId = " + hostID + ", connectionId = "
            + connectionID + ", error = " + error.ToString() + ")");
    }


    void ServerButton()
    {
        byte error;
        //Open the sockets for sending and receiving the messages on port 54321
        m_ServerSocket = NetworkTransport.AddHost(m_HostTopology, 54321);
        //Connect the "server"
        NetworkTransport.Connect(m_ServerSocket, "127.0.0.1", 54321, 0, out erro
        //Check for if there is an error
        if ((NetworkError)error != NetworkError.Ok)
        {
            //Output this message in the console with the Network Error
            Debug.Log("There was this error : " + (NetworkError)error);
        }
        //Otherwise if no errors occur, output this message to the console
        else Debug.Log("Connected : " + (NetworkError)error);
    }
}
```

Properties

Ok	The operation completed successfully.
WrongHost	The specified host not available.
WrongConnection	The specified connectionId doesn't exist.
WrongChannel	The specified channel doesn't exist.
NoResources	Not enough resources are available to process this request.
BadMessage	Not a data message.
Timeout	Connection timed out.
MessageToLong	The message is too long to fit the buffer.
WrongOperation	Operation is not supported.

VersionMismatch	The protocol versions are not compatible. Check your library versions.
CRCMismatch	The Networking.ConnectionConfig does not match the other endpoint.
DNSFailure	The address supplied to connect to was invalid or could not be resolved.
UsageError	This error will occur if any function is called with inappropriate parameter values.

Description

Event that is returned when calling the Networking.NetworkTransport.Receive and Networking.NetworkTransport.ReceiveFromHost functions.

Properties

DataEvent	Data event received. Indicating that data was received.
ConnectEvent	Connection event received. Indicating that a new connection was established.
DisconnectEvent	Disconnection event received.
Nothing	No new event was received.
BroadcastEvent	Broadcast discovery event received. To obtain sender connection info and possible complimentary message from them, call Networking.NetworkTransport.GetBroadcastConnectionInfo() and Networking.NetworkTransport.GetBroadcastConnectionMessage() functions.

Description

Enumeration of methods of where to spawn player objects in multiplayer games.

```
using UnityEngine;
using UnityEngine.Networking;


public class PlayerSpawnMethodExample : MonoBehaviour
{
    void Update()
    {
        //Press the space key to switch to spawning on a random spawn point
        if (Input.GetKeyDown(KeyCode.Space))
        {
            //Check that the PlayerSpawnMethod is currently RoundRobin
            if (NetworkManager.singleton.playerSpawnMethod == PlayerSpawnMethod.
                //Switch it to Random spawning if it is
                NetworkManager.singleton.playerSpawnMethod = PlayerSpawnMethod.R
            //Otherwise switch it to RoundRobin
            else NetworkManager.singleton.playerSpawnMethod = PlayerSpawnMethod.
        }
    }
}
```

Properties

Random	Spawn players at a randomly chosen starting position.
RoundRobin	Spawn players at the next start position.

Description

Enumeration of all supported quality of service channel modes.

Properties

Unreliable	There is no guarantee of delivery or ordering.
UnreliableFragmented	There is no guarantee of delivery or ordering, but allowing fragmented messages with up to 32 fragments per message.
UnreliableSequenced	There is no guarantee of delivery and all unordered messages will be dropped. Example: VoIP.
Reliable	Each message is guaranteed to be delivered but not guaranteed to be in order.
ReliableFragmented	Each message is guaranteed to be delivered, also allowing fragmented messages with up to 32 fragments per message.
ReliableSequenced	Each message is guaranteed to be delivered and in order.
StateUpdate	An unreliable message. Only the last message in the send buffer is sent. Only the most recent message in the receive buffer will be delivered.
ReliableStateUpdate	A reliable message. Note: Only the last message in the send buffer is sent. Only the most recent message in the receive buffer will be delivered.
AllCostDelivery	A reliable message that will be re-sent with a high frequency until it is acknowledged.
UnreliableFragmentedSequenced	There is garantee of ordering, no guarantee of delivery, but allowing fragmented messages with up to 32 fragments per message.
ReliableFragmentedSequenced	Each message is guaranteed to be delivered in order, also allowing fragmented messages with up to 32 fragments per message.

Description

Define how unet will handle network io operation.

Properties

SelectReactor	Network thread will sleep up to threadawake timeout, or up to receive event on socket will happened. Awaked thread will try to read up to maxpoolsize packets from socket and will try update connections ready to send (with fixing awaketimeout rate).
FixRateReactor	Network thread will sleep up to threadawake timeout, after that it will try receive up to maxpoolsize amount of messages and then will try perform send operation for connection whihc ready to send.

Description

Enumeration of Networking versions.

Properties

Current	The current UNET version.

Obsolete

Description

A Custom Attribute that can be added to member functions of NetworkBehaviour scripts, to make them only run on clients.

A [Client] method returns immediately if NetworkClient.active is not true, and generates a warning on the console. This attribute can be put on member functions that are meant to be only called on clients. This would redundant for [ClientRPC] functions, as being client-only is already enforced for them.

```
using UnityEngine;
using UnityEngine.Networking;


public class Example : MonoBehaviour
{
    [Client]
    public void OnClientDisconnected(NetworkConnection conn, NetworkReader reade
    {
        Debug.Log("Client Disconnected");
        //ShutdownGame();
        Application.LoadLevel("title");
    }
}
```

Obsolete

Description

A Custom Attribute that can be added to member functions of NetworkBehaviour scripts, to make them only run on clients, but not generate warnings.

This custom attribute is the same as the Client custom attribute, except that it does not generate a warning in the console if called on a server. This is useful to avoid spamming the console for functions that will be invoked by the engine, such as Update() or physics callbacks.

```
using UnityEngine;
using UnityEngine.Networking;


public class Example : MonoBehaviour
{
    [ClientCallback]
    void OnTriggerEnter2D(Collider2D collider)
    {
        // make explosion
    }
}
```

This will make the explosion code only run when the trigger is hit on the client.

Obsolete

Description

This is an attribute that can be put on methods of NetworkBehaviour classes to allow them to be invoked on clients from a server.

[ClientRPC] functions are called by code on Unity Multiplayer servers, and then invoked on corresponding GameObjects on clients connected to the server. The arguments to the RPC call are serialized across the network, so that the client function is invoked with the same values as the function on the server. These functions must begin with the prefix "Rpc" and cannot be static.

```
using UnityEngine;
using UnityEngine.Networking;


public class Example : NetworkBehaviour
{
    int counter;
    [ClientRpc]
    public void RpcDoMagic(int extra)
    {
        Debug.Log("Magic = " + (123 + extra));
    }


    void Update()
    {
        counter += 1;


        if (counter % 100 == 0 && NetworkServer.active)
        {
            RpcDoMagic(counter);
        }
    }
}
```

The allowed argument types are;

- Basic type (byte, int, float, string, UInt64, etc)
- Built-in Unity math type (Vector3, Quaternion, etc),
- Arrays of basic types

- Structs containing allowable types
- NetworkIdentity
- NetworkInstanceId
- NetworkHash128
- GameObject with a NetworkIdentity component attached.

Obsolete

Description

This is an attribute that can be put on methods of NetworkBehaviour classes to allow them to be invoked on the server by sending a command from a client.

[Command] functions are invoked on the player GameObject associated with a connection. This is set up in response to the "ready" message, by passing the player GameObject to the NetworkServer.PlayerIsReady() function. The arguments to the command call are serialized across the network, so that the server function is invoked with the same values as the function on the client. These functions must begin with the prefix "Cmd" and cannot be static.

```
using UnityEngine;
using UnityEngine.Networking;



public class Player : NetworkBehaviour
{
    int moveX = 0;
    int moveY = 0;
    float moveSpeed = 0.2f;
    bool isDirty = false;


    void Update()
    {
        if (!isLocalPlayer)
        {
            return;
        }


        // input handling for local player only
        int oldMoveX = moveX;
        int oldMoveY = moveY;


        moveX = 0;
        moveY = 0;


        if (Input.GetKey(KeyCode.LeftArrow))
```

```
        {
            moveX -= 1;
        }
        if (Input.GetKey(KeyCode.RightArrow))
        {
            moveX += 1;
        }
        if (Input.GetKey(KeyCode.UpArrow))
        {
            moveY += 1;
        }
        if (Input.GetKey(KeyCode.DownArrow))
        {
            moveY -= 1;
        }
        if (moveX != oldMoveX || moveY != oldMoveY)
        {
            CmdMove(moveX, moveY);
        }
    }


    [Command]
    public void CmdMove(int x, int y)
    {
        moveX = x;
        moveY = y;
        isDirty = true;
    }


    public void FixedUpdate()
    {
        if (NetworkServer.active)
        {
            transform.Translate(moveX * moveSpeed, moveY * moveSpeed, 0);
        }
    }
}
```

The allowed argument types are;

- Basic type (byte, int, float, string, UInt64, etc)
- Built-in Unity math type (Vector3, Quaternion, etc),
- Arrays of basic types

- Structs containing allowable types
- NetworkIdentity
- NetworkInstanceId
- NetworkHash128
- GameObject with a NetworkIdentity component attached.

Obsolete

Description

This attribute is used to configure the network settings of scripts that are derived from the NetworkBehaviour base class.

```
using UnityEngine.Networking;


[NetworkSettings(channel = 1, sendInterval = 0.2f)]
class MyScript : NetworkBehaviour
{
    [SyncVar]
    int value;
}
```

Obsolete

Description

A Custom Attribute that can be added to member functions of NetworkBehaviour scripts, to make them only run on servers.

A [Server] method returns immediately if NetworkServer.active is not true, and generates a warning on the console. This attribute can be put on member functions that are meant to be only called on server. This would be redundant for Command functions, as being server-only is already enforced for them.

```
using UnityEngine;
using UnityEngine.Networking;


public class Example : NetworkBehaviour
{
    [Server]
    public void Explode()
    {
        NetworkServer.Destroy(gameObject);
    }
}
```

Obsolete

Description

A Custom Attribute that can be added to member functions of NetworkBehaviour scripts, to make them only run on servers, but not generate warnings.

This custom attribute is the same as the [Server] custom attribute, except that it does not generate a warning in the console if called on a client. This is useful to avoid spamming the console for functions that will be invoked by the engine, such as Update() or physics callbacks.

```
using UnityEngine;
using UnityEngine.Networking;


public class Example : MonoBehaviour
{
    float regenTimer = 0;
    int heat = 100;


    [ServerCallback]
    void Update()
    {
        // heat dissipates over time
        if (Time.time > regenTimer)
        {
            if (heat > 1)
                heat -= 2;


            regenTimer = Time.time + 1.0f;
        }
    }
}
```

Obsolete

Description

This is an attribute that can be put on events in NetworkBehaviour classes to allow them to be invoked on client when the event is called on the server.

[SyncEvent] events are called by user code on UNET servers, and then invoked on corresponding client objects on clients connected to the server. The arguments to the Event call are serialized across the network, so that the client event is invoked with the same values as the function on the server. These events must begin with the prefix "Event".

```
using UnityEngine;
using UnityEngine.Networking;


public class DamageClass : NetworkBehaviour
{
    public delegate void TakeDamageDelegate(int amount, float dir);


    [SyncEvent]
    public event TakeDamageDelegate EventTakeDamage;


    [Command]
    public void CmdDoMe(int val)
    {
        EventTakeDamage(val, 1.0f);
    }
}


public class Other : NetworkBehaviour
{
    public DamageClass damager;
    int health = 100;


    void Start()
    {
        if (NetworkClient.active)
            damager.EventTakeDamage += TakeDamage;
    }
```

```
    public void TakeDamage(int amount, float dir)
    {
        health -=  amount;
    }
}
```

SyncEvents allow networked actions to be propagated to other scripts attached to the object. In the example above, the Other class registers for the TakeDamage event on the DamageClass. When the event happens on the DamageClass on the server, the TakeDamage() method will be invoked on the Other class on the client object. This allows modular network aware systems to be created, that can be extended by new scripts that respond to the events generated by them.

Obsolete

Description

[SyncVar] is an attribute that can be put on member variables of NetworkBehaviour classes. These variables will have their values sychronized from the server to clients in the game that are in the ready state.

Setting the value of a [SyncVar] marks it as dirty, so it will be sent to clients at the end of the current frame. Only simple values can be marked as [SyncVars]. The type of the SyncVar variable cannot be from an external DLL or assembly.

```
using UnityEngine;
using UnityEngine.Networking;


public class Ship : NetworkBehaviour
{
    [SyncVar]
    public int health = 100;


    [SyncVar]
    public float energy = 100;
}
```

The allowed SyncVar types are;

- Basic type (byte, int, float, string, UInt64, etc)
- Built-in Unity math type (Vector3, Quaternion, etc),
- Structs containing allowable types .

Obsolete

Description

This is an attribute that can be put on methods of NetworkBehaviour classes to allow them to be invoked on clients from a server. Unlike the ClientRpc attribute, these functions are invoked on one individual target client, not all of the ready clients.

[TargetRpc] functions are called by user code on the server, and then invoked on the corresponding client object on the client of the specified NetworkConnection. The arguments to the RPC call are serialized across the network, so that the client function is invoked with the same values as the function on the server. These functions must begin with the prefix "Target" and cannot be static.

The first argument to an TargetRpc function must be a NetworkConnection object.

```
using UnityEngine;
using UnityEngine.Networking;


public class Example : NetworkBehaviour
{
    [TargetRpc]
    public void TargetDoMagic(NetworkConnection target, int extra)
    {
        Debug.Log("Magic = " + (123 + extra));
    }


    [Command]
    void CmdTest()
    {
        TargetDoMagic(connectionToClient, 55);
    }
}
```

The allowed argument types are;

- Basic type (byte, int, float, string, UInt64, etc)
- Built-in Unity math type (Vector3, Quaternion, etc),
- Arrays of basic types
- Structs containing allowable types
- NetworkIdentity
- NetworkInstanceId

- NetworkHash128
- GameObject with a NetworkIdentity component attached.

Description

Implements high-level utility methods to simplify use of the Playable API with Animations.

Static Methods

Play	Plays the Playable on the given Animator.
PlayAnimatorController	Creates a PlayableGraph to be played on the given Animator. An AnimatorControllerPlayable is also created for the given RuntimeAnimatorController.
PlayClip	Creates a PlayableGraph to be played on the given Animator. An AnimationClipPlayable is also created for the given AnimationClip.
PlayLayerMixer	Creates a PlayableGraph to be played on the given Animator. An AnimationLayerMixerPlayable is also created.
PlayMixer	Creates a PlayableGraph to be played on the given Animator. An AnimationMixerPlayable is also created.

Description

This structure contains the frame information a Playable receives in Playable.PrepareFrame.

Properties

deltaTime	Time difference between this frame and the preceding frame.
effectiveParentDelay	The accumulated delay of the parent Playable during the PlayableGraph traversal.
effectiveParentSpeed	The accumulated speed of the parent Playable during the PlayableGraph traversal.
effectiveSpeed	The accumulated speed of the Playable during the PlayableGraph traversal.
effectiveWeight	The accumulated weight of the Playable during the PlayableGraph traversal.
evaluationType	Indicates the type of evaluation that caused PlayableGraph.PrepareFrame to be called.
frameId	The current frame identifier.
output	The PlayableOutput that initiated this graph traversal.
seekOccurred	Indicates that the local time was explicitly set.
timeHeld	Indicates the local time did not advance because it has reached the duration and the extrapolation mode is set to Hold.
timeLooped	Indicates the local time wrapped because it has reached the duration and the extrapolation mode is set to Loop.
weight	The weight of the current Playable.

Description

Describes the cause for the evaluation of a PlayableGraph.

Properties

Evaluate	Indicates the graph was updated due to a call to PlayableGraph.Evaluate.
Playback	Indicates the graph was called by the runtime during normal playback due to PlayableGraph.Play being called.

Implements interfaces:IPlayable

Description

Playables are customizable runtime objects that can be connected together and are contained in a PlayableGraph to create complex behaviours.

Playables can be used to create complex and flexible data evaluation trees. Playables are nodes that can be connected together, after which each Playable can set the "weight" or "influence" of each of its children.

The playables of the same graph are contained in a PlayableGraph. A PlayableGraph can have several outputs, also called "players", which implement IPlayableOutput. The PlayableOutput takes the result of their source playable and apply it to an object in the Scene. For instance, the AnimationPlayableOutput is linked to a playable node in the graph (the "source playable") and to an Animator in the Scene. When the graph is played, the animation pose resulting of the graph evaluation is applied by the Animator. There are as many PlayableOutputs as there are different playable types: AnimationPlayableOutput, AudioPlayableOutput, TexturePlayableOutput, ScriptPlayableOutput, etc...

The ScriptPlayable<T> is a special kind of playable. It's main role is to be a "custom" playable. It is a templated struct where `T` must derived from PlayableBehaviour. These custom PlayableBehaviours allow to write behaviours at specific moments in the graph evaluation (see PlayableBehaviour.PrepareFrame and PlayableBehaviour.ProcessFrame). A good example of a ScriptPlayable is the TimelinePlayable which is controlling the Timeline graph. It creates and links together the playables in charge of the tracks and the clips.

When a PlayableGraph is played, each PlayableOutput will be traversed. During this traversal, it will call the PrepareFrame method on each Playable. This allows the Playable to "prepare itself for the next evaluation". It is during the PrepareFrame stage that each Playable can modify its children (either by adding new inputs or by removing some of them). This enables Playable to "spawn" new children branches in the Playable tree at runtime. This means that Playable trees are not static structures. They can adapt and change over time.

Once the preparation is done, the PlayableOutputs are in charge of processing the result, that's why they are also called "players". In the case of an AnimationPlayableOutput, the Animator is in charge of processing the graph. And in the case of a ScriptPlayableOutput, PlayableBehaviour.ProcessFrame will be called on each ScriptPlayable.

Note: You can use the PlayableExtensions methods on any struct implementing IPlayable.

Note: The Manual has detailed documentation about the Playables API.

Static Properties

Null	Returns an invalid Playable.

Implements interfaces:IPlayableAsset

Description

A base class for assets that can be used to instantiate a Playable at runtime.

Properties

duration	The playback duration in seconds of the instantiated Playable.
outputs	A description of the outputs of the instantiated Playable.

Public Methods

CreatePlayable	Implement this method to have your asset inject playables into the given graph.

Inherited Members

Properties

hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.

Public Methods

GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.

Static Methods

Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.

CreateInstance	Creates an instance of a scriptable object.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Messages

Awake	This function is called when the ScriptableObject script is started.
OnDestroy	This function is called when the scriptable object will be destroyed.
OnDisable	This function is called when the scriptable object goes out of scope.
OnEnable	This function is called when the object is loaded.

Implements interfaces:IPlayableBehaviour

Description

PlayableBehaviour is the base class from which every custom playable script derives.

A PlayableBehaviour can be used to add user-defined behaviour to a PlayableGraph.

A PlayableBehaviour must be part of a branch of a PlayableGraph that is connected to an output to be active.

In the following example, two AnimationClip are controlled by two AnimationClipPlayable, which are blended by a AnimationMixerPlayable. A custom BlenderPlayableBehaviour is modifying the inputs weigth of the AnimationMixerPlayable every frame.

```
using UnityEngine;
using UnityEngine.Animations;
using UnityEngine.Playables;


public class BlenderPlayableBehaviour : PlayableBehaviour
{
    public AnimationMixerPlayable mixerPlayable;


    public override void PrepareFrame(Playable playable, FrameData info)
    {
        float blend = Mathf.PingPong((float)playable.GetTime(), 1.0f);


        mixerPlayable.SetInputWeight(0, blend);
        mixerPlayable.SetInputWeight(1, 1.0f - blend);


        base.PrepareFrame(playable, info);
    }
}


public class PlayableBehaviourSample : MonoBehaviour
{
    PlayableGraph m_Graph;
    public AnimationClip clipA;
    public AnimationClip clipB;
```

```
// Use this for initialization
void Start()
{
    // Create the PlayableGraph.
    m_Graph = PlayableGraph.Create();


    // Add an AnimationPlayableOutput to the graph.
    var animOutput = AnimationPlayableOutput.Create(m_Graph, "AnimationOutpu


    // Add an AnimationMixerPlayable to the graph.
    var mixerPlayable = AnimationMixerPlayable.Create(m_Graph, 2, false);


    // Add two AnimationClipPlayable to the graph.
    var clipPlayableA = AnimationClipPlayable.Create(m_Graph, clipA);
    var clipPlayableB = AnimationClipPlayable.Create(m_Graph, clipB);


    // Add a custom PlayableBehaviour to the graph.
    // This behavior will change the weights of the mixer dynamically.
    var blenderPlayable = ScriptPlayable<BlenderPlayableBehaviour>.Create(m_
    blenderPlayable.GetBehaviour().mixerPlayable = mixerPlayable;


    // Create the topology, connect the AnimationClipPlayable to the
    // AnimationMixerPlayable.  Also add the BlenderPlayableBehaviour.
    m_Graph.Connect(clipPlayableA, 0, mixerPlayable, 0);
    m_Graph.Connect(clipPlayableB, 0, mixerPlayable, 1);
    m_Graph.Connect(mixerPlayable, 0, blenderPlayable, 0);


    // Use the ScriptPlayable as the source for the AnimationPlayableOutput.
    // Since it's a ScriptPlayable, also set the source input port to make t
    // passthrough to the AnimationMixerPlayable.
    animOutput.SetSourcePlayable(blenderPlayable);
    animOutput.SetSourceInputPort(0);


    // Play the graph.
    m_Graph.Play();
}
```

```
    private void OnDestroy()
    {
        // Destroy the graph once done with it.
        m_Graph.Destroy();
    }
}
```

Public Methods

OnBehaviourDelay	This function is called when the Playable play state is changed to PlayState.Delayed.
OnBehaviourPause	This function is called when the Playable play state is changed to PlayState.Paused.
OnBehaviourPlay	This function is called when the Playable play state is changed to PlayState.Playing.
OnGraphStart	This function is called when the PlayableGraph that owns this PlayableBehaviour starts.
OnGraphStop	This function is called when the PlayableGraph that owns this PlayableBehaviour stops.
OnPlayableCreate	This function is called when the Playable that owns the PlayableBehaviour is created.
OnPlayableDestroy	This function is called when the Playable that owns the PlayableBehaviour is destroyed.
PrepareData	This function is called during the PrepareData phase of the PlayableGraph.
PrepareFrame	This function is called during the PrepareFrame phase of the PlayableGraph.
ProcessFrame	This function is called during the ProcessFrame phase of the PlayableGraph.

Description

Struct that holds information regarding an output of a PlayableAsset.

PlayableAssets specify the type of outputs it supports using PlayableBindings.

Do not create PlayableBinding objects directly. Use the provided built-in methods to create the corresponding PlayableOutput. For example, to create a PlayableBinding for an AnimationPlayableOutput, use AnimationPlayableBinding.Create. To create a PlayableBinding for a ScriptPlayableOutput, use ScriptPlayableBinding.Create.

Static Properties

DefaultDuration	The default duration used when a PlayableOutput has no fixed duration.
None	A constant to represent a PlayableAsset has no bindings.

Properties

outputTargetType	The type of target required by the PlayableOutput for this PlayableBinding.
sourceObject	A reference to a UnityEngine.Object that acts a key for this binding.
streamName	The name of the output or input stream.

Leave feedback

Implements interfaces:IExposedPropertyTable

Description

Instantiates a PlayableAsset and controls playback of Playable objects.

Properties

duration	The duration of the Playable in seconds.
extrapolationMode	Controls how the time is incremented when it goes beyond the duration of the playable.
initialTime	The time at which the Playable should start when first played.
playableAsset	The PlayableAsset that is used to instantiate a playable for playback.
playableGraph	The PlayableGraph created by the PlayableDirector.
playOnAwake	Whether the playable asset will start playing back as soon as the component awakes.
state	The current playing state of the component. (Read Only)
time	The component's current time. This value is incremented according to the PlayableDirector.timeUpdateMode when it is playing. You can also change this value manually.
timeUpdateMode	Controls how time is incremented when playing back.

Public Methods

ClearGenericBinding	Clears the binding of a reference object.
ClearReferenceValue	Clears an exposed reference value.
DeferredEvaluate	Tells the PlayableDirector to evaluate it's PlayableGraph on the next update.
Evaluate	Evaluates the currently playing Playable at the current time.
GetGenericBinding	Returns a binding to a reference object.
GetReferenceValue	Retreives an ExposedReference binding.
Pause	Pauses playback of the currently running playable.
Play	Instatiates a Playable using the provided PlayableAsset and starts playback.
RebindPlayableGraphOutputs	Rebinds each PlayableOutput of the PlayableGraph.
RebuildGraph	Discards the existing PlayableGraph and creates a new instance.

Resume	Resume playing a paused playable.
SetGenericBinding	Sets the binding of a reference object from a PlayableBinding.
SetReferenceValue	Sets an ExposedReference value.
Stop	Stops playback of the current Playable and destroys the corresponding graph.

Events

paused	Event that is raised when a PlayableDirector component has paused.
played	Event that is raised when a PlayableDirector component has begun playing.
stopped	Event that is raised when a PlayableDirector component has stopped.

Inherited Members

Properties

enabled	Enabled Behaviours are Updated, disabled Behaviours are not.
isActiveAndEnabled	Has the Behaviour had active and enabled called?
gameObject	The game object this component is attached to. A component is always attached to a game object.
tag	The tag of this game object.
transform	The Transform attached to this GameObject.
hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.

Public Methods

BroadcastMessage	Calls the method named methodName on every MonoBehaviour in this game object or any of its children.
CompareTag	Is this game object tagged with tag ?
GetComponent	Returns the component of Type type if the game object has one attached, null if it doesn't.
GetComponentInChildren	Returns the component of Type type in the GameObject or any of its children using depth first search.
GetComponentInParent	Returns the component of Type type in the GameObject or any of its parents.

GetComponents	Returns all components of Type type in the GameObject.
GetComponentsInChildren	Returns all components of Type type in the GameObject or any of its children.
GetComponentsInParent	Returns all components of Type type in the GameObject or any of its parents.
SendMessage	Calls the method named methodName on every MonoBehaviour in this game object.
SendMessageUpwards	Calls the method named methodName on every MonoBehaviour in this game object and on every ancestor of the behaviour.
GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.

Static Methods

Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Description

Extensions for all the types that implements IPlayable.

Extension methods are static methods that can be called as if they were instance methods on the extended type.

```
using UnityEngine;
using UnityEngine.Animations;
using UnityEngine.Playables;


public class ExamplePlayableBehaviour : PlayableBehaviour
{
    void Start()
    {
        PlayableGraph graph = PlayableGraph.Create();
        AnimationMixerPlayable mixer = AnimationMixerPlayable.Create(graph, 1);


        // Calling method PlayableExtensions.SetDuration on AnimationMixerPlayab
        mixer.SetDuration(10);


        // The line above is the same as calling directly PlayableExtensions.Set
        PlayableExtensions.SetDuration(mixer, 10);
    }
}
```

Static Methods

AddInput	Create a new input port and connect it to the output port of the given Playable.
ConnectInput	Connect the output port of a Playable to one of the input ports.
Destroy	Destroys the current Playable.
DisconnectInput	Disconnect the input port of a Playable.
GetDelay	Returns the delay of the playable.
GetDuration	Returns the duration of the Playable.
GetGraph	Returns the PlayableGraph that owns this Playable. A Playable can only be used in the

	graph that was used to create it.
GetInput	Returns the Playable connected at the given input port index.
GetInputCount	Returns the number of inputs supported by the Playable.
GetInputWeight	Returns the weight of the Playable connected at the given input port index.
GetLeadTime	Returns the Playable lead time in seconds.
GetOutput	Returns the Playable connected at the given output port index.
GetOutputCount	Returns the number of outputs supported by the Playable.
GetPlayState	Returns the current PlayState of the Playable.
GetPreviousTime	Returns the previous local time of the Playable.
GetPropagateSetTime	Returns the time propagation behavior of this Playable.
GetSpeed	Returns the speed multiplier that is applied to the the current Playable.
GetTime	Returns the current local time of the Playable.
GetTraversalMode	Returns the propagation mode for the multi-output playable.
IsDelayed	Returns whether or not the Playable has a delay.
IsDone	Returns a flag indicating that a playable has completed its operation.
IsNull	Returns true if the Playable is null, false otherwise.
IsValid	Returns the vality of the current Playable.
Pause	Tells to pause the Playable.
Play	Starts to play the Playable.
SetDelay	Set a delay until the playable starts.
SetDone	Changes a flag indicating that a playable has completed its operation.
SetDuration	Changes the duration of the Playable.
SetInputCount	Changes the number of inputs supported by the Playable.
SetInputWeight	Changes the weight of the Playable connected to the current Playable.
SetLeadTime	Sets the Playable lead time in seconds.
SetOutputCount	Changes the number of outputs supported by the Playable.
SetPropagateSetTime	Changes the time propagation behavior of this Playable.
SetSpeed	Changes the speed multiplier that is applied to the the current Playable.

SetTime	Changes the current local time of the Playable.
SetTraversalMode	Sets the propagation mode of PrepareFrame and ProcessFrame for the multi-output playable.

Description

Use the PlayableGraph to manage Playable creations and destructions.

The PlayableGraph is also the link to different systems, through structs that implement IPlayableOutput. For example, AnimationPlayableOutput or AudioPlayableOutput.

Public Methods

Connect	Connects two Playable instances.
Destroy	Destroys the graph.
DestroyOutput	Destroys the PlayableOutput.
DestroyPlayable	Destroys the Playable.
DestroySubgraph	Destroys the Playable and all its inputs, recursively.
Disconnect	Disconnects the Playable. The connections determine the topology of the PlayableGraph and how it is evaluated.
Evaluate	Evaluates all the PlayableOutputs in the graph, and updates all the connected Playables in the graph.
GetEditorName	Returns the name of the PlayableGraph.
GetOutput	Get PlayableOutput at the given index in the graph.
GetOutputByType	Get PlayableOutput of the requested type at the given index in the graph.
GetOutputCount	Returns the number of PlayableOutput in the graph.
GetOutputCountByType	Get the number of PlayableOutput of the requested type in the graph.
GetPlayableCount	Returns the number of Playable owned by the Graph.
GetResolver	Returns the table used by the graph to resolve ExposedReferences.
GetRootPlayable	Returns the Playable with no output connections at the given index.
GetRootPlayableCount	Returns the number of Playable owned by the Graph that have no connected outputs.
GetTimeUpdateMode	Returns how time is incremented when playing back.
IsDone	Indicates that a graph has completed its operations.
IsPlaying	Indicates that a graph is presently running.

IsValid	Returns true if the PlayableGraph has been properly constructed using PlayableGraph.CreateGraph and is not deleted.
Play	Plays the graph.
SetResolver	Changes the table used by the graph to resolve ExposedReferences.
SetTimeUpdateMode	Changes how time is incremented when playing back.
Stop	Stops the graph, if it is playing.

Static Methods

Create	Creates a PlayableGraph.

Implements interfaces:IPlayableOutput

Description

See: IPlayableOutput.

Static Properties

Null	Returns an invalid PlayableOutput.

Description

Extensions for all the types that implements IPlayableOutput.

Extension methods are static methods that can be called as if they were instance methods on the extended type.

```
using UnityEngine;
using UnityEngine.Playables;


public class ExamplePlayableBehaviour : PlayableBehaviour
{
    void Start()
    {
        PlayableGraph graph = PlayableGraph.Create();
        ScriptPlayableOutput scriptOutput = ScriptPlayableOutput.Create(graph, "


        // Calling method PlayableExtensions.SetWeight on ScriptPlayableOutput a
        scriptOutput.SetWeight(10);


        // The line above is the same as calling directly PlayableExtensions.Set
        PlayableOutputExtensions.SetWeight(scriptOutput, 10);
    }
}
```

Static Methods

AddNotificationReceiver	Registers a new receiver that listens for notifications.
GetNotificationReceivers	Retrieves the list of notification receivers currently registered on the output.
GetSourceOutputPort	Returns the source playable's output connection index.
GetSourcePlayable	Returns the source playable.
GetUserData	Returns the opaque user data. This is the same value as the last last argument of ProcessFrame.
GetWeight	Returns the weight of the connection from the PlayableOutput to the source playable.

IsOutputNull	Returns true if the PlayableOutput is null, false otherwise.
IsOutputValid	
PushNotification	Queues a notification to be sent through the Playable system.
RemoveNotificationReceiver	Unregisters a receiver on the output.
SetReferenceObject	Sets the bound object to a new value. Used to associate an output to an object (Track asset in case of Timeline).
SetSourceOutputPort	Sets the source playable's output connection index. For playables with multiple outputs, this determines which sub-branch of the source playable generates this output.
SetSourcePlayable	Sets which playable that computes the output and which sub-tree index.
SetUserData	Sets the opaque user data. This same data is passed as the last argument to ProcessFrame.
SetWeight	Sets the weight of the connection from the PlayableOutput to the source playable.

Implements interfaces:IPlayable

Description

A IPlayable implementation that contains a PlayableBehaviour for the PlayableGraph. PlayableBehaviour can be used to write custom Playable that implement their own PrepareFrame callback.

A branch of a PlayableGraph must be connected to an output to be evaluated.

NOTE: You can use PlayableExtensions methods with ScriptPlayable objects.

Description

A PlayableBinding that contains information representing a ScriptingPlayableOutput.

Static Methods

Create	Creates a PlayableBinding that contains information representing a ScriptPlayableOutput.

Implements interfaces:IPlayableOutput

Description

A IPlayableOutput implementation that contains a script output for the a PlayableGraph.

A branch of a PlayableGraph must be connected to an output to be evaluated.

NOTE: You can use PlayableOutputExtensions methods with ScriptPlayableOutput objects.

Static Properties

Null	Returns an invalid ScriptPlayableOutput.

Static Methods

Create	Creates a new ScriptPlayableOutput in the associated PlayableGraph.

Description

Interface implemented by all C# Playable implementations.

NOTE: You can use PlayableExtensions methods with all objects that implement IPlayable.

Description

Interface that permits a class to inject playables into a graph.

Properties

duration	Duration in seconds.
outputs	A description of the PlayableOutputs generated by this asset.

Public Methods

CreatePlayable	Implement this method to have your asset inject playables into the given graph.

Description

Interface implemented by all C# Playable Behaviour implementations.

Description

Interface implemented by all C# Playable output implementations.

NOTE: You can use PlayableOutputExtensions methods with all objects that implement IPlayableOutput.

Obsolete

Description

Describes the type of information that flows in and out of a Playable. This also specifies that this Playable is connectable to others of the same type.

Description

Defines what time source is used to update a Director graph.

Properties

DSPClock	Update is based on DSP (Digital Sound Processing) clock. Use this for graphs that need to be synchronized with Audio.
GameTime	Update is based on Time.time. Use this for graphs that need to be synchronized on gameplay, and that need to be paused when the game is paused.
UnscaledGameTime	Update is based on Time.unscaledTime. Use this for graphs that need to be updated even when gameplay is paused. Example: Menus transitions need to be updated even when the game is paused.
Manual	Update mode is manual. You need to manually call PlayableGraph.Evaluate with your own deltaTime. This can be useful for graphs that are completely disconnected from the rest of the game. For example, localized bullet time.

Description

Wrap mode for Playables.

Properties

Hold	Hold the last frame when the playable time reaches it's duration.
Loop	Loop back to zero time and continue playing.
None	Do not keep playing when the time reaches the duration.

Description

Traversal mode for Playables.

Properties

Mix	Causes the Playable to prepare and process it's inputs when demanded by an output.
Passthrough	Causes the Playable to act as a passthrough for PrepareFrame and ProcessFrame. If the PlayableOutput being processed is connected to the n-th input port of the Playable, the Playable only propagates the n-th output port. Use this enum value in conjunction with PlayableOutput SetSourceOutputPort.

Description

Status of a Playable.

Properties

Paused	The Playable has been paused. Its local time will not advance.
Playing	The Playable is currently Playing.
Delayed	The Playable has been delayed, using PlayableExtensions.SetDelay. It will not start until the delay is entirely consumed.

Description

Custom CPU Profiler label used for profiling arbitrary code blocks.

Use CustomSampler to measure execution time of script code blocks. Produced information is displayed in the CPU Profiler and can be captured with Recorder.
Using CustomSampler is more efficient than using Profiler.BeginSample to profile your code. This is because CustomSamplers that have been created in advance have very low Begin call overhead compared to Profiler.BeginSample.

```
using UnityEngine;
using UnityEngine.Profiling;


public class ExampleClass : MonoBehaviour
{
    CustomSampler sampler;
    void Start()
    {
        sampler = CustomSampler.Create("MyCustomSampler");
    }


    void Update()
    {
        sampler.Begin();
        // do something that takes a lot of time
        sampler.End();
    }
}
```

CustomSampler.Begin is conditionally compiled away using ConditionalAttribute. Thus it will have zero overhead, when it is deployed in non-Development Build.

See Also: Sampler, CustomSampler.Create, CustomSampler.Begin.

Public Methods

Begin	Begin profiling a piece of code with a custom label defined by this instance of CustomSampler.
End	End profiling a piece of code with a custom label.

Static Methods

Create	Creates a new CustomSampler for profiling parts of your code.

Inherited Members

Properties

isValid	Returns true if Sampler is valid. (Read Only)
name	Sampler name. (Read Only)

Public Methods

GetRecorder	Returns Recorder associated with the Sampler.

Static Methods

Get	Returns Sampler object for the specific CPU Profiler label.
GetNames	Returns number and names of all registered Profiler labels.

Description

Controls the Profiler from script.

You can add custom Profiler sections in your scripts with Profiler.BeginSample and Profiler.EndSample.

On standalone platforms, you can save all profiling information to a file, which allows you to inspect it later. To do this, you must specify a Profiler.logFile and set both Profiler.enabled and Profiler.enableBinaryLog to `true`.

Because use of the Profiler negatively affects the performance of your app, most of the Profiler API functionality is only available when "Development Build" is enabled. This means you must enable "Developer Build" if you want to use profiler API methods in your built app. Disabling "Development Build" makes your app run faster, but prevents you from using most of the Profiler API methods.

The exception to this are the Profiler API methods relating to memory usage. Because Unity manages most of its system memory at run-time, it can provide that information with no performance penalty, therefore those methods are available even if "Development Build" is not enabled. This applies to all memory-related Profiler API methods *except* Profiler.GetAllocatedMemoryForGraphicsDriver and Profiler.GetRuntimeMemorySizeLong, since they require extra profiling data only available in development builds.

Static Properties

areaCount	The number of Profiler Areas that you can profile.
enableBinaryLog	Enables the logging of profiling data to a file.
enabled	Enables the Profiler.
logFile	Specifies the file to use when writing profiling data.
maxUsedMemory	Sets the maximum amount of memory that Profiler uses for buffering data. This property is expressed in bytes.
usedHeapSizeLong	Returns the number of bytes that Unity has allocated. This does not include bytes allocated by external libraries or drivers.

Static Methods

AddFramesFromFile	Displays the recorded profile data in the profiler.
BeginSample	Begin profiling a piece of code with a custom label.
BeginThreadProfiling	Enables profiling on the thread from which you call this method.
EndSample	Ends the current profiling sample.

EndThreadProfiling	Frees the internal resources used by the Profiler for the thread.
GetAllocatedMemoryForGraphicsDriver	Returns the amount of allocated memory for the graphics driver, in bytes.Only available in development players and editor.
GetAreaEnabled	Returns whether or not a given ProfilerArea is currently enabled.
GetMonoHeapSizeLong	Returns the size of the reserved space for managed-memory.
GetMonoUsedSizeLong	The allocated managed-memory for live objects and non-collected objects.
GetRuntimeMemorySizeLong	Gathers the native-memory used by a Unity object.
GetTempAllocatorSize	Returns the size of the temp allocator.
GetTotalAllocatedMemoryLong	The total memory allocated by the internal allocators in Unity. Unity reserves large pools of memory from the system. This function returns the amount of used memory in those pools.
GetTotalReservedMemoryLong	The total memory Unity has reserved.
GetTotalUnusedReservedMemoryLong	Unity allocates memory in pools for usage when unity needs to allocate memory. This function returns the amount of unused memory in these pools.
SetAreaEnabled	Enable or disable a given ProfilerArea.
SetTempAllocatorRequestedSize	Sets the size of the temp allocator.

Description

Records profiling data produced by a specific Sampler.

Recorder accumulates elapsed time of every Begin/End pair during a frame. Once enabled it sums delta time between Begin and End samples produced by a Sampler. Use elapsedNanoseconds to get the accumulated time for the previous frame.

```
using UnityEngine;
using UnityEngine.Profiling;


public class ExampleClass : MonoBehaviour
{
    Recorder behaviourUpdateRecorder;
    void Start()
    {
        behaviourUpdateRecorder = Recorder.Get("BehaviourUpdate");
        behaviourUpdateRecorder.enabled = true;
    }


    void Update()
    {
        if (behaviourUpdateRecorder.isValid)
            Debug.Log("BehaviourUpdate time: " + behaviourUpdateRecorder.elapsed
    }
}
```

Note: A Recorder collects data regardless of the Profiler state. When a Recorder is enabled, data is accumulated every time its Sampler is hit. This information is equivalent to the one you can see in Hierarchy view in the Profiler Window.

At the moment Samplers are available only in the Editor and Development Players. Use Recorder.isValid to verify if Recorder can collect the data.

Recorder supports only internal static Profiler labels and labels generated by CustomSampler. Dynamic internal labels produced by scripting method calls and labels produced by Profiler.BeginSample are not supported.

See Also: Sampler, Recorder.isValid, Recorder.elapsedNanoseconds.

Properties

elapsedNanoseconds	Accumulated time of Begin/End pairs for the previous frame in nanoseconds. (Read Only)
enabled	Enables recording.
isValid	Returns true if Recorder is valid and can collect data. (Read Only)
sampleBlockCount	Number of time Begin/End pairs was called during the previous frame. (Read Only)

Public Methods

CollectFromAllThreads	Configures the recorder to collect samples from all threads.
FilterToCurrentThread	Configures the recorder to only collect data from the current thread.

Static Methods

Get	Use this function to get a Recorder for the specific Profiler label.

Description

Provides control over a CPU Profiler label.

Sampler is a counter which produces timings information you can see in CPU Profiler. Use this class to get information about built-in or custom Profiler label.

See Also: Sampler.Get, CustomSampler, CPU Usage Profiler.

Properties

isValid	Returns true if Sampler is valid. (Read Only)
name	Sampler name. (Read Only)

Public Methods

GetRecorder	Returns Recorder associated with the Sampler.

Static Methods

Get	Returns Sampler object for the specific CPU Profiler label.
GetNames	Returns number and names of all registered Profiler labels.

Implements interfaces:IPurchasingModule

Description

Helper base class for IAP Modules.

In addition to providing helper methods, use of an abstract class such as this allows addition of IPurchasingModule methods without breaking compatibility with existing plugins.

Public Methods

Configure	Called when your module is loaded by Unity.

Protected Methods

BindConfiguration	Bind a store-specific configuration type to a concrete instance.
BindExtension	Bind a store-specific extension type to a concrete instance.
RegisterStore	Register a store implementation along with its name.

Implements interfaces:IStore

Description

Extension point for purchasing plugins.

An abstract class is provided so that methods can be added to the IStore without breaking binary compatibility with existing plugins.

See Also: IStore.

Public Methods

FinishTransaction	Called when Unity IAP has finished processing a purchase.
Initialize	Called when Unity IAP is initializing.
Purchase	Called when a user wants to buy the specified Product.
RetrieveProducts	Fetch product metadata and purchase state for the set of ProductDefinitions.

Description

A common format which store subsystems use to describe available products.

Includes purchase state via `receipt` and `transactionId`.

Properties

metadata	Localized metadata retrieved from the Store.
receipt	A purchase receipt, if owned. Otherwise null.
storeSpecificId	The store-specific ID.
transactionId	The purchase transaction ID, if owned. Otherwise null.
type	The ProductType.

Constructors

ProductDescription	Create a ProductDescription.

Description

Represents a failed purchase as described by a purchasing service.

Is sent by a store implementation to IAP which forwards this message on to the user.
See Also: IStoreCallback.OnPurchaseFailed.

Properties

message	More information about the purchase failure from Unity IAP or the platform store, if available.
productId	The store-specific product ID which failed to purchase.
reason	The reason for the purchase failure.

Constructors

PurchaseFailureDescription	Creates a PurchaseFailureDescription.

Description

ICatalogProvider can replace or add products to the product configuration at runtime.

When Purchasing is initialized, if the ConfigurationBuilder has the useCatalogProvider property set to true, the ICatalogProvider's FetchProducts method will be called before the initialization is completed.

Public Methods

FetchProducts	ProductDefinitions fetched by this method will add to or replace the set of ProductDefinitions supplied in the ConfigurationBuilder.

Description

Configures Unity IAP with one or more store implementations.

Passed into IPurchasingModules during Unity IAP initialization.

See Also: IPurchasingModule.

Public Methods

RegisterConfiguration	Informs Unity IAP that extended Configuration is available.
RegisterExtension	Informs Unity IAP that a store extension is available.
RegisterStore	Informs Unity IAP that a store implementation exists, specifying its name.
SetCatalogProvider	Sets the active ICatalogProvider. Calling this method after Purchasing is initialized has no effect.

Description

A Unity IAP configuration of one or more stores.

Store implementations must provide an implementation of this interface. Use IPurchasingModule implementations with the `ConfigurationBuilder` to extend Unity IAP's store functionality.

```
using UnityEngine;


class MyPurchasingModule : IPurchasingModule
{
    public void Configure(IPurchasingBinder binder)
    {
        binder.RegisterStore("MyManufacturerAppStore", InstantiateMyManufacturer
        // Our Purchasing service implementation provides the real implementatio
        binder.RegisterExtension<IManufacturerExtensions>(new FakeManufacturerEx
    }


    IStore InstantiateMyManufacturerAppStore()
    {
        // Check for Manufacturer. "Android" used here for the sake of example.
        // In your implementation, return platform-appropriate store instead of
        if (Application.platform == RuntimePlatform.Android) { return null; }
        else { return null; }
    }


    IStoreExtension IManufacturerExtensions() { return null; }
}
```

See Also: AbstractPurchasingModule, IStore, IStoreExtension, ConfigurationBuilder.

Public Methods

Configure	Called when Unity IAP is loading your module. Register stores and associated extensions using the IPurchasingBinder.

Description

The public interface of an underlying store system (e.g. Google Play or Apple App Store) typically exposed to Unity IAP extending its in-app purchasing platform support.

Unity IAP is extensible, supporting registration of store systems through `IPurchasingModule` implementations shared with Unity IAP via a `ConfigurationBuilder` during initialization.

A sample store class:

```
using System;
using System.Collections.ObjectModel;
using System.Collections.Generic;
using UnityEngine.Purchasing;
using UnityEngine.Purchasing.Extension;


// Purchases always succeed at the sample store
internal class SampleStore : IStore
{
    public const string Name = "samplestore";
    private IStoreCallback m_Biller;
    private List<string> m_PurchasedProducts = new List<string>();


    public void Initialize(IStoreCallback biller)
    {
        m_Biller = biller;
    }


    public void RetrieveProducts(ReadOnlyCollection<ProductDefinition> productDe
    {
        var products = new List<ProductDescription>();
        foreach (var product in productDefinitions)
        {
            var metadata = new ProductMetadata("$123.45", "Fake title for " + pr
            products.Add(new ProductDescription(product.storeSpecificId, metadat
        }
        m_Biller.OnProductsRetrieved(products);
    }
```

```
    public void Purchase(ProductDefinition product, string developerPayload)
    {
        // Keep track of non consumables.
        if (product.type != ProductType.Consumable)
        {
            m_PurchasedProducts.Add(product.storeSpecificId);
        }
        m_Biller.OnPurchaseSucceeded(product.storeSpecificId, "{ \"this\" : \"is
    }


    public void FinishTransaction(ProductDefinition product, string transactionI
    {
    }
}
```

See Also: AbstractStore, UnityPurchasing.Initialize, IPurchasingModule, ConfigurationBuilder.

Public Methods

FinishTransaction	Called by Unity IAP when a transaction has been recorded.
Initialize	Initialize the store.
Purchase	Handle a purchase request from a user.
RetrieveProducts	Fetch the latest product metadata, including purchase receipts, asynchronously with results returned via IStoreCallback.

Description

Callback used by store implementations to interact with Unity IAP.

Properties

products	Gets the item with local identifier.
useTransactionLog	Toggle use of Unity IAP's transaction log.

Public Methods

OnProductsRetrieved	Complete setup by providing a list of available products, complete with metadata and any associated purchase receipts and transaction IDs.
OnPurchaseFailed	Call to indicate to Unity IAP that a purchase failed.
OnPurchaseSucceeded	Inform Unity IAP of a purchase.
OnSetupFailed	Indicate that IAP is unavailable for a specific reason, such as IAP being disabled in device settings.

Description

Common interface for all extended configuration of stores.

Use during Unity IAP configuration by `IPurchasingModule` before initialization. Bound to store during `IPurchasingModule` configuration via `ConfigurationBuilder`.

Sample store configuration interface (needing implementation), to be instantiated by the time the owning purchasing module is requested to configure its stores:

```
using UnityEngine.Purchasing.Extension;


// Sample store config. Implementation can be used by the user,
// and accessed by the owning purchasing module during store
// instantiation.
public interface ISampleConfiguration : IStoreConfiguration
{
    // Request access to sample debugging backend
    bool useSampleStoreDebuggingBackend { get; set; }
}
```

See Also: IPurchasingModule, ConfigurationBuilder.

Description

Builds configurations for Unity IAP.

Collects products and store-specific configuration details.

See Also: UnityPurchasing.Initialize.

Properties

products	The products built so far.
useCatalogProvider	If this property is set to true, Purchasing will use the supplied ICatalogProvider to augment the supplied set of products during initialization.

Public Methods

AddProduct	Add a product with a Unity IAP ID, type and optional set of store-specific IDs.
AddProducts	Add a collection of ProductDefinitions.
Configure	Access store-specific Configuration settings.

Static Methods

Instance	Get an instance of ConfigurationBuilder.

Description

Maps store-specific product identifiers to one or more store names.

See Also: ConfigurationBuilder.AddProduct.

Constructors

IDs	Create a new mapping of store identifiers to store names.

Public Methods

Add	Add a product ID which is supported by a list of store platform names.
GetEnumerator	Enumerator for IDs.

Description

Defines a reward that is paid to the player when an in-app purchase product is purchased.

Static Properties

MaxDataLength	Represents the maximum allowed length for the data field on a PayoutDefinition.
MaxSubtypeLength	Represents the maximum allowed length for the subtype field on a PayoutDefinition.

Properties

data	A field for user-specified data.
quantity	The quantity of this payout.
subtype	The subtype of this payout.
type	The type of this payout.
typeString	A string representation of the type of this payout.

Constructors

PayoutDefinition	Create a PayoutDefinition and define the values for the type, subtype, quantity, and data fields.

Description

Represents a product that may be purchased as an In-App Purchase.

Here is an example of a sample Store class working with a Product references.

```
using System;
using UnityEngine;
using UnityEngine.Purchasing;


public class MyStoreClass : MonoBehaviour, IStoreListener
{
    static string kProductID100Currency = "virtualcurrency_100";
    IStoreController m_StoreController;
    void Awake()
    {
        ConfigurationBuilder builder = ConfigurationBuilder.Instance(StandardPur
        builder.AddProduct(kProductID100Currency, ProductType.Consumable);
        UnityPurchasing.Initialize(this, builder);
    }


    public void PurchaseCurrency()
    {
        if (m_StoreController != null)
        {
            // Fetch the currency Product reference from Unity Purchasing
            Product product = m_StoreController.products.WithID(kProductID100Cur
            if (product != null && product.availableToPurchase)
            {
                m_StoreController.InitiatePurchase(product);
            }
        }
    }


    public void OnInitialized(IStoreController controller, IExtensionProvider ex
    {
        m_StoreController = controller;
    }
```

```
    public void OnInitializeFailed(InitializationFailureReason error) {}
    public PurchaseProcessingResult ProcessPurchase(PurchaseEventArgs e)
    {
        if (String.Equals(e.purchasedProduct.definition.id, kProductID100Currenc
        {
            Debug.Log("Purchased 100 coins");
        }
        return PurchaseProcessingResult.Complete;
    }


    public void OnPurchaseFailed(Product item, PurchaseFailureReason r) {}
}
```

Properties

availableToPurchase	Determine if this product is available to purchase according to the store subsystem.
definition	Fundamental immutable product properties.
hasReceipt	Owned Non Consumables and Subscriptions should always have receipts.
metadata	Localized metadata provided by the store system.
receipt	The purchase receipt for this product, if owned. Otherwise null.
transactionID	A unique identifier for this product's transaction, if available. Otherwise null.

Public Methods

Equals	Equality defined for use in collections.
GetHashCode	GetHashCode defined for use in collections.

Description

Provides helper methods to retrieve products by ID.

Supports retrieving by both store-independent and store-specific product ID. Retrieves products registered with Unity IAP.

Properties

all	All products.
set	The set of products.

Public Methods

WithID	Get the product with store-independent Unity IAP ID.
WithStoreSpecificID	Get the product with the store-specific ID.

Description

Minimal product definition, used by apps declaring products for sale.

Properties

enabled	This flag indicates whether a product should be offered for sale. It is controlled through the cloud catalog dashboard.
id	Unity IAP product ID. Potentially independent of store IDs.
payout	Access the first payout on this ProductDefinition. Returns null if there are no payouts specified.
payouts	Access all payouts attached to this ProductDefinition.
storeSpecificId	The ID this product has on a store.
type	The product type.

Constructors

ProductDefinition	Create a ProductDefinition with different Store-independent ID and Store-specific ID. Use this when you need to two IDs to be different.

Public Methods

Equals	Compares id properties. Requires obj be a ProductDefinition.
GetHashCode	Gets the application-domain-specific hash code of id.

Description

Localized information about a product, retrieved from a store.

Properties

isoCurrencyCode	Product currency in ISO 4217 format; e.g. GBP or USD.
localizedDescription	Localized product description as retrieved from the store subsystem; e.g. Apple or Google.
localizedPrice	Decimal product price denominated in the currency indicated by isoCurrencySymbol.
localizedPriceString	Localized price string.
localizedTitle	Localized product title as retrieved from the store subsystem; e.g. Apple or Google.

Constructors

ProductMetadata	Create a ProductMetadata.

Description

A purchase that succeeded, including the purchased product along with its purchase receipt.

See Also: IStoreListener.ProcessPurchase.

Properties

purchasedProduct	The Product that was purchased.

Description

A purchase failed event containing diagnostic data.

Explains the purchase failure.

Properties

message	More information about the purchase failure, if available. Otherwise null.
purchasedProduct	The product that failed to purchase.
reason	The reason for the purchase failure.

Description

Entry point for Applications using Unity IAP.

Class is available to compiler and runtime when Unity IAP is enabled in Editor's Services window.

Static Methods

ClearTransactionLog	Clears Unity IAP's internal transaction log.
Initialize	Initialize Unity IAP with the specified listener and configuration.

Description

Provides access to store-specific extended functionality.

Specific extension's interface type can be associated with a specific store type via instantiating `IPurchasingModule`'s `IPurchasingBinder`.

Sample extension:

```
using System;
using UnityEngine;
using UnityEngine.Purchasing;


// Extends an implementing store.
public interface ISampleExtension : IStoreExtension
{
    // Implemented by associated store.
    void RestoreTransactions(Action<bool> callback);
}
```

See Also: IStoreExtension, IPurchasingModule, IPurchasingBinder.

Public Methods

GetExtension	Get the store-specific extension of specified type.

Description

Used by Applications to control Unity IAP.

See Also: Product for an example usage.

Properties

products	Store products including metadata and purchase receipts.

Public Methods

ConfirmPendingPurchase	Confirm a pending purchase.
FetchAdditionalProducts	This method may be used to fetch additional products for purchasing, or to refresh metadata on existing products.
InitiatePurchase	Initiate a purchase for a specific product.

Description

Common interface for all purchasing extensions.

See Also: IExtensionProvider.

Description

Notifies your Application of purchasing related events.

See Also: Product for an example implementation.

Public Methods

OnInitialized	Called when Unity IAP has retrieved all product metadata and is ready to make purchases.
OnInitializeFailed	Note that Unity IAP will not call this method if the device is offline, but continually attempt initialization until online.
OnPurchaseFailed	Called when a purchase fails.
ProcessPurchase	Called when a purchase succeeds.

Description

The various reasons Unity IAP initialization can fail.

See Also: IStoreListener.OnInitializeFailed.

Properties

PurchasingUnavailable	In-App Purchases disabled in device settings.
NoProductsAvailable	No products available for purchase.
AppNotKnown	The store reported the app as unknown.

Description

Specifies the type of a PayoutDefinition.

Properties

Other	Use this for payouts that are not covered by the other values in the PayoutType enum.
Currency	Use this for payouts that grant currency to the player, such as gold or gems.
Resource	Use this for payouts that grant the player a resource like health or food.

Description

Categories of purchasable product.

See Also: ConfigurationBuilder.AddProduct, ConfigurationBuilder.AddProducts, and ProductDefinition.type.

Properties

Consumable	Can be purchased repeatedly. Suitable for consumable products such as virtual currencies.
NonConsumable	Can only be purchased once. Suitable for one-off purchases such as extra levels.
Subscription	Can be purchased repeatedly and restored. Durable but with a finite duration of validity.

Description

The various reasons a purchase can fail.

See Also: IStoreListener.OnPurchaseFailed.

Properties

PurchasingUnavailable	The system purchasing feature is unavailable.
ExistingPurchasePending	A purchase was already in progress when a new purchase was requested.
ProductUnavailable	The product is not available to purchase on the store.
SignatureInvalid	Signature validation of the purchase's receipt failed.
UserCancelled	The user opted to cancel rather than proceed with the purchase.
PaymentDeclined	There was a problem with the payment.
DuplicateTransaction	A duplicate transaction error when the transaction has already been completed successfully.
Unknown	A catch-all for unrecognized purchase problems.

Description

Informs Unity IAP as to whether an Application has finished processing a purchase.

Use to complete or delay completion of acknowledging receipt of purchase.

See Also: IStoreListener.ProcessPurchase.

Properties

Complete	The application has finished processing the purchase.
Pending	The application has not finished processing the purchase, e.g. it is pushing it to a server asynchronously.

Description

Allows the asynchronous read back of GPU resources.

This class is used to copy resource data from the GPU to the CPU without any stall (GPU or CPU), but adds a few frames of latency. See Also: AsyncGPUReadbackRequest.

Static Methods

Request	Triggers a request to asynchronously fetch the data from a GPU resource.

Description

Represents an asynchronous request for a GPU resource.

Use AsyncGPUReadback.Request to retrieve an asynchronous request for a GPU resource. Pending requests are automatically updated each frame. The result is accessible only for a single frame once is successfully fulfilled and this request is then disposed of in the following frame. Common uses for this are to query AsyncGPUReadbackRequest.done each frame (or within a coroutine) and then call AsyncGPUReadbackRequest.GetData if the AsyncGPUReadbackRequest.hasError is false. You don't have to manage the request lifetime as this is managed internally. A request that has been disposed of will result in the AsyncGPUReadbackRequest.hasError property being true. See Also:AsyncGPUReadback.

Properties

depth	When reading data from a ComputeBuffer, depth is 1, otherwise, the property takes the value of the requested depth from the texture.
done	Checks whether the request has been processed.
hasError	This property is true if the request has encountered an error.
height	When reading data from a ComputeBuffer, height is 1, otherwise, the property takes the value of the requested height from the texture.
layerCount	Number of layers in the current request.
layerDataSize	The size in bytes of one layer of the readback data.
width	The width of the requested GPU data.

Public Methods

GetData	Fetches the data of a successful request.
Update	Triggers an update of the request.
WaitForCompletion	Waits for completion of the request.

Description

List of graphics commands to execute.

Command buffers hold list of rendering commands ("set render target, draw mesh, ..."). They can be set to execute at various points during camera rendering (see Camera.AddCommandBuffer), light rendering (see Light.AddCommandBuffer) or be executed immediately (see Graphics.ExecuteCommandBuffer).

Typically they would be used to extend Unity's rendering pipeline in some custom ways. For example, you could render some additional objects into deferred rendering g-buffer after all regular objects are done, or do custom processing of light shadow maps. See command buffers overview page for more details.

Command buffers can be created and then executed many times if needed.

See Also: Camera.AddCommandBuffer, Light.AddCommandBuffer, CameraEvent, LightEvent, Graphics.ExecuteCommandBuffer, command buffers overview.

Properties

name	Name of this command buffer.
sizeInBytes	Size of this command buffer in bytes (Read Only).

Constructors

CommandBuffer	Create a new empty command buffer.

Public Methods

BeginSample	Adds a command to begin profile sampling.
Blit	Add a "blit into a render texture" command.
Clear	Clear all commands in the buffer.
ClearRandomWriteTargets	Clear random write targets for Shader Model 4.5 level pixel shaders.
ClearRenderTarget	Adds a "clear render target" command.
ConvertTexture	Converts and copies a source texture to a destination texture with a different format or dimensions.
CopyCounterValue	Adds a command to copy ComputeBuffer counter value.
CopyTexture	Adds a command to copy a texture into another texture.

CreateGPUFence	Creates a GPUFence which will be passed after the last Blit, Clear, Draw, Dispatch or Texture Copy command prior to this call has been completed on the GPU.
DisableScissorRect	Add a command to disable the hardware scissor rectangle.
DisableShaderKeyword	Adds a command to disable global shader keyword.
DispatchCompute	Add a command to execute a ComputeShader.
DrawMesh	Add a "draw mesh" command.
DrawMeshInstanced	Add a "draw mesh with instancing" command.The command will not immediately fail and throw an exception if Material.enableInstancing is false, but it will log an error and skips rendering each time the command is being executed if such a condition is detected.InvalidOperationException will be thrown if the current platform doesn't support this API (i.e. if GPU instancing is not available). See SystemInfo.supportsInstancing.
DrawMeshInstancedIndirect	Add a "draw mesh with indirect instancing" command.
DrawProcedural	Add a "draw procedural geometry" command.
DrawProceduralIndirect	Add a "draw procedural geometry" command.
DrawRenderer	Add a "draw renderer" command.
EnableScissorRect	Add a command to enable the hardware scissor rectangle.
EnableShaderKeyword	Adds a command to enable global shader keyword.
EndSample	Adds a command to begin profile sampling.
GenerateMips	Generate mipmap levels of a render texture.
GetTemporaryRT	Add a "get a temporary render texture" command.
GetTemporaryRTArray	Add a "get a temporary render texture array" command.
IssuePluginCustomBlit	Send a user-defined blit event to a native code plugin.
IssuePluginCustomTextureUpdateV2	Send a texture update event to a native code plugin.
IssuePluginEvent	Send a user-defined event to a native code plugin.
IssuePluginEventAndData	Send a user-defined event to a native code plugin with custom data.
ReleaseTemporaryRT	Add a "release a temporary render texture" command.
RequestAsyncReadback	Adds an asynchonous GPU readback request command to the command buffer.

ResolveAntiAliasedSurface	Force an antialiased render texture to be resolved.
SetComputeBufferParam	Adds a command to set an input or output buffer parameter on a ComputeShader.
SetComputeFloatParam	Adds a command to set a float parameter on a ComputeShader.
SetComputeFloatParams	Adds a command to set multiple consecutive float parameters on a ComputeShader.
SetComputeIntParam	Adds a command to set an integer parameter on a ComputeShader.
SetComputeIntParams	Adds a command to set multiple consecutive integer parameters on a ComputeShader.
SetComputeMatrixArrayParam	Adds a command to set a matrix array parameter on a ComputeShader.
SetComputeMatrixParam	Adds a command to set a matrix parameter on a ComputeShader.
SetComputeTextureParam	Adds a command to set a texture parameter on a ComputeShader.
SetComputeVectorArrayParam	Adds a command to set a vector array parameter on a ComputeShader.
SetComputeVectorParam	Adds a command to set a vector parameter on a ComputeShader.
SetGlobalBuffer	Add a "set global shader buffer property" command.
SetGlobalColor	Add a "set global shader color property" command.
SetGlobalDepthBias	Add a command to set global depth bias.
SetGlobalFloat	Add a "set global shader float property" command.
SetGlobalFloatArray	Add a "set global shader float array property" command.
SetGlobalInt	Sets the given global integer property for all shaders.
SetGlobalMatrix	Add a "set global shader matrix property" command.
SetGlobalMatrixArray	Add a "set global shader matrix array property" command.
SetGlobalTexture	Add a "set global shader texture property" command, referencing a RenderTexture.
SetGlobalVector	Add a "set global shader vector property" command.
SetGlobalVectorArray	Add a "set global shader vector array property" command.
SetInvertCulling	Add a "set invert culling" command to the buffer.
SetProjectionMatrix	Add a command to set the projection matrix.

SetRandomWriteTarget	Set random write target for Shader Model 4.5 level pixel shaders.
SetRenderTarget	Add a "set active render target" command.
SetShadowSamplingMode	Add a "set shadow sampling mode" command.
SetViewMatrix	Add a command to set the view matrix.
SetViewport	Add a command to set the rendering viewport.
SetViewProjectionMatrices	Add a command to set the view and projection matrices.
WaitOnGPUFence	Instructs the GPU to wait until the given GPUFence is passed.

Description

Used to manage synchronisation between tasks on async compute queues and the graphics queue.

Not all platforms support GPU fences. See SystemInfo.supportsGPUFence.

A GPUFence represents a point during GPU processing after a specific compute shader dispatch or draw call has completed. It can be used to achieve synchronisation between tasks running on the async compute queues or the graphics queue by having one or more queues wait until a given fence has passed. This is an important consideration when working with async compute as which tasks are running at the same time on the graphics queue and the async compute queues is key to improving GPU performance.

GPUFence's do not need to be used to synchronise where one GPU task is writing to a resource that will be read as an input by another. These resource dependencies are automatically handled by Unity.

GPUFences should be created via Graphics.CreateGPUFence or CommandBuffer.CreateGPUFence attempting to use a GPUFence that has not been created via one of these functions will result in an exception.

It is possible to create circular dependencies using GPUFences that, if executed, would deadlock the GPU. Unity will detect such circular dependencies in the Editor and raise exceptions if any exist after calls to Graphics.CreateGPUFence, Graphics.WaitOnGPUFence, Graphics.ExecuteCommandBuffer, Graphics.ExecuteCommandBufferAsync, ScriptableRenderContext.ExecuteCommandBuffer, ScriptableRenderContext.ExecuteCommandBufferAsync.

See Also: SystemInfo.supportsGPUFence, Graphics.CreateGPUFence, Graphics.WaitOnGPUFence, CommandBuffer.CreateGPUFence, CommandBuffer.WaitOnGPUFence, Graphics.ExecuteCommandBuffer, Graphics.ExecuteCommandBufferAsync, ScriptableRenderContext.ExecuteCommandBuffer, ScriptableRenderContext.ExecuteCommandBufferAsync.

Properties

passed	Has the GPUFence passed?Allows for CPU determination of whether the GPU has passed the point in its processing represented by the GPUFence.

Description

Script interface for Graphics Settings.

Static Properties

lightsUseColorTemperature	Whether to use a Light's color temperature when calculating the final color of that Light."
lightsUseLinearIntensity	If this is true, Light intensity is multiplied against linear color values. If it is false, gamma color values are used.
renderPipelineAsset	The RenderPipelineAsset that describes how the Scene should be rendered.
transparencySortAxis	An axis that describes the direction along which the distances of objects are measured for the purpose of sorting.
transparencySortMode	Transparent object sorting mode.
useScriptableRenderPipelineBatching	Enable/Disable SRP batcher (experimental) at runtime.

Static Methods

GetCustomShader	Get custom shader used instead of a built-in shader.
GetShaderMode	Get built-in shader mode.
HasShaderDefine	Returns true if shader define was set when compiling shaders for current GraphicsTier.
SetCustomShader	Set custom shader to use instead of a built-in shader.
SetShaderMode	Set built-in shader mode.

Inherited Members

Properties

hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.

Public Methods

GetInstanceID	Returns the instance id of the object.

ToString	Returns the name of the GameObject.

Static Methods

Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Description

A collection of ShaderKeyword that represents a specific platform variant.

See Also: ShaderKeyword, Rendering.ShaderKeywordSetUtility, Shader.EnableKeyword, Shader.EnableKeyword, multiple variants.

Public Methods

Disable	Disable a specific shader keyword.
Enable	Enable a specific shader keyword.
IsEnabled	Check whether a specific shader keyword is enabled.

Description

ReflectionProbeBlendInfo contains information required for blending probes.

See Also: Renderer.reflectionProbeUsage.

Properties

probe	Reflection Probe used in blending.
weight	Specifies the weight used in the interpolation between two probes, value varies from 0.0 to 1.0.

Description

Describes a render target with one or more color buffers, a depth/stencil buffer and the associated load/store-actions that are applied when the render target is active.

This data structure is similiar to RenderTargetSetup, but relies on a RenderTargetIdentifier to ensure compatability with CommandBuffer.SetRenderTarget.

To render to a specific mip-level, cubemap face or depth slice the RenderTargetIdentifier should be created accordingly.

Note: the number of load- and store-actions specified for color buffers must be equal to the number of color buffers.

See Also: CommandBuffer.

Properties

colorLoadActions	Load actions for color buffers.
colorRenderTargets	Color buffers to use as render targets.
colorStoreActions	Store actions for color buffers.
depthLoadAction	Load action for the depth/stencil buffer.
depthRenderTarget	Depth/stencil buffer to use as render target.
depthStoreAction	Store action for the depth/stencil buffer.

Constructors

RenderTargetBinding	Constructs RenderTargetBinding.

Description

Identifies a RenderTexture for a CommandBuffer.

Render textures can be identified in a number of ways, for example a RenderTexture object, or one of built-in render textures (BuiltinRenderTextureType), or a temporary render texture with a name (that was created using CommandBuffer.GetTemporaryRT).

This struct serves as a way to identify them, and has implcit conversion operators so that in most cases you can save some typing.

See Also: CommandBuffer.

Constructors

RenderTargetIdentifier	Creates a render target identifier.

Description

Identifier of a specific code path in a shader.

See Also: ShaderKeywordSet, Rendering.ShaderKeywordSetUtility, Shader.EnableKeyword, multiple variants.

Constructors

ShaderKeyword	Initializes a new instance of the ShaderKeyword class from a shader keyword name.

Public Methods

GetKeywordName	Returns the string name of the keyword.
GetKeywordType	Returns the keyword kind: built-in or user defined.
IsValid	Returns true if the keyword has been imported by Unity.

Description

A collection of ShaderKeyword that represents a specific shader variant.

See Also: ShaderKeyword, Rendering.ShaderKeywordSetUtility, Shader.EnableKeyword, Shader.EnableKeyword, multiple variants.

Public Methods

Disable	Disable a specific shader keyword.
Enable	Enable a specific shader keyword.
GetShaderKeywords	Return an array with all the enabled keywords in the ShaderKeywordSet.
IsEnabled	Check whether a specific shader keyword is enabled.

Description

Adding a SortingGroup component to a GameObject will ensure that all Renderers within the GameObject's descendants will be sorted and rendered together.

A common use case for having a SortingGroup is to create complex 2D characters that are made up of multiple SpriteRenderers. When several clones of such a character overlap, their individual body parts might not be sorted properly resulting in a visual glitch where the the body parts interleave. For example, the hands of two characters might be sorted in front of their bodies, where you would expect one entire character to be drawn in front of the other character. The SortingGroup component solves this by ensuring the entire branch of the character are sorted and rendered together.

The descendants of the SortingGroup are sorted using the same SortingLayer and Renderer.sortingOrder. However, they are only sorted against other descendants of the SortingGroup and not with any renderers outside of it. This allows you to reuse the same SortingLayers (for example, "Hands", "Torso"...) to sort body parts while ensuring they never interleave with other clones of the character.

The SortingGroups, together with other renderers, are sorted using the SortingLayer and Renderer.sortingOrder. Additionally, they can be nested within other SortingGroups. This is useful if you have branches of descendants that should not be mixed up i.e. the "Left Hand" vs the "Right Hand" hierarchy branches.

.

Properties

sortingLayerID	Unique ID of the Renderer's sorting layer.
sortingLayerName	Name of the Renderer's sorting layer.
sortingOrder	Renderer's order within a sorting layer.

Inherited Members

Properties

enabled	Enabled Behaviours are Updated, disabled Behaviours are not.
isActiveAndEnabled	Has the Behaviour had active and enabled called?
gameObject	The game object this component is attached to. A component is always attached to a game object.
tag	The tag of this game object.
transform	The Transform attached to this GameObject.
hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?

name	The name of the object.

Public Methods

BroadcastMessage	Calls the method named methodName on every MonoBehaviour in this game object or any of its children.
CompareTag	Is this game object tagged with tag ?
GetComponent	Returns the component of Type type if the game object has one attached, null if it doesn't.
GetComponentInChildren	Returns the component of Type type in the GameObject or any of its children using depth first search.
GetComponentInParent	Returns the component of Type type in the GameObject or any of its parents.
GetComponents	Returns all components of Type type in the GameObject.
GetComponentsInChildren	Returns all components of Type type in the GameObject or any of its children.
GetComponentsInParent	Returns all components of Type type in the GameObject or any of its parents.
SendMessage	Calls the method named methodName on every MonoBehaviour in this game object.
SendMessageUpwards	Calls the method named methodName on every MonoBehaviour in this game object and on every ancestor of the behaviour.
GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.

Static Methods

Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Description

Spherical harmonics up to the second order (3 bands, 9 coefficients).

Spherical harmonics (SH) represent a function or signal over directions, and are commonly used in computer graphics to efficiently evaluate smooth lighting. Unity uses them for LightProbes and environment lighting.

L2 spherical harmonics is composed of 9 coefficients for each color channel.

See Also: RenderSettings.ambientMode, RenderSettings.ambientProbe, LightProbes.

Properties

this[int,int]	Access individual SH coefficients.

Public Methods

AddAmbientLight	Add ambient lighting to probe data.
AddDirectionalLight	Add directional light to probe data.
Clear	Clears SH probe to zero.
Evaluate	Evaluates the Spherical Harmonics for each of the given directions. The result from the first direction is written into the first element of results, the result from the second direction is written into the second element of results, and so on. The array size of directions and results must match and directions must be normalized.

Operators

operator !=	Returns true if SH probes are different.
operator *	Scales SH by a given factor.
operator +	Adds two SH probes.
operator ==	Returns true if SH probes are equal.

Description

Provides an interface to the Unity splash screen.

Static Properties

isFinished	Returns true once the splash screen as finished. This is once all logos have been shown for their specified duration.

Static Methods

Begin	Initializes the splash screen so it is ready to begin drawing. Call this before you start calling SplashScreen.Draw. Internally this function resets the timer and prepares the logos for drawing.
Draw	Immediately draws the splash screen. Ensure you have called SplashScreen.Begin before you start calling this.

Description

Ambient lighting mode.

Unity can provide ambient lighting in several modes, for example directional ambient with separate sky, equator and ground colors, or flat ambient with a single color.

See Also: RenderSettings.ambientMode, Lighting Window.

Properties

Skybox	Skybox-based or custom ambient lighting.
Trilight	Trilight ambient lighting.
Flat	Flat ambient lighting.
Custom	Ambient lighting is defined by a custom cubemap.

Description

Blend mode for controlling the blending.

The blend mode is set separately for source and destination, and it controls the blend factor of each component going into the blend equation. It is also possible to set the blend mode for color and alpha components separately. Note: the blend modes are ignored if logical blend operations or advanced OpenGL blend operations are in use.

Properties

Zero	Blend factor is (0, 0, 0, 0).
One	Blend factor is (1, 1, 1, 1).
DstColor	Blend factor is (Rd, Gd, Bd, Ad).
SrcColor	Blend factor is (Rs, Gs, Bs, As).
OneMinusDstColor	Blend factor is (1 - Rd, 1 - Gd, 1 - Bd, 1 - Ad).
SrcAlpha	Blend factor is (As, As, As, As).
OneMinusSrcColor	Blend factor is (1 - Rs, 1 - Gs, 1 - Bs, 1 - As).
DstAlpha	Blend factor is (Ad, Ad, Ad, Ad).
OneMinusDstAlpha	Blend factor is (1 - Ad, 1 - Ad, 1 - Ad, 1 - Ad).
SrcAlphaSaturate	Blend factor is (f, f, f, 1); where f = min(As, 1 - Ad).
OneMinusSrcAlpha	Blend factor is (1 - As, 1 - As, 1 - As, 1 - As).

Description

Blend operation.

The blend operation that is used to combine the pixel shader output with the render target. This can be passed through Material.SetInt() to change the blend operation during runtime.

Note that the logical operations are only supported in Gamma (non-sRGB) colorspace, on DX11.1 hardware running on DirectX 11.1 runtime.

Advanced OpenGL blend operations are supported only on hardware supporting either GL_KHR_blend_equation_advanced or GL_NV_blend_equation_advanced and may require use of GL.RenderTargetBarrier. In addition, the shaders that are used with the advanced blend operations must have a UNITY_REQUIRE_ADVANDED_BLEND(mode) declaration in the shader code where mode is one of the blend operations or "all_equations" for supporting all advanced blend operations (see the KHR_blend_equation_advanced spec for other values).

Properties

Add	Add (s + d).
Subtract	Subtract.
ReverseSubtract	Reverse subtract.
Min	Min.
Max	Max.
LogicalClear	Logical Clear (0).
LogicalSet	Logical SET (1) (D3D11.1 only).
LogicalCopy	Logical Copy (s) (D3D11.1 only).
LogicalCopyInverted	Logical inverted Copy (!s) (D3D11.1 only).
LogicalNoop	Logical No-op (d) (D3D11.1 only).
LogicalInvert	Logical Inverse (!d) (D3D11.1 only).
LogicalAnd	Logical AND (s & d) (D3D11.1 only).
LogicalNand	Logical NAND !(s & d). D3D11.1 only.
LogicalOr	Logical OR (s | d) (D3D11.1 only).
LogicalNor	Logical NOR !(s | d) (D3D11.1 only).

LogicalXor	Logical XOR (s XOR d) (D3D11.1 only).
LogicalEquivalence	Logical Equivalence !(s XOR d) (D3D11.1 only).
LogicalAndReverse	Logical reverse AND (s & !d) (D3D11.1 only).
LogicalAndInverted	Logical inverted AND (!s & d) (D3D11.1 only).
LogicalOrReverse	Logical reverse OR (s | !d) (D3D11.1 only).
LogicalOrInverted	Logical inverted OR (!s | d) (D3D11.1 only).
Multiply	Multiply (Advanced OpenGL blending).
Screen	Screen (Advanced OpenGL blending).
Overlay	Overlay (Advanced OpenGL blending).
Darken	Darken (Advanced OpenGL blending).
Lighten	Lighten (Advanced OpenGL blending).
ColorDodge	Color dodge (Advanced OpenGL blending).
ColorBurn	Color burn (Advanced OpenGL blending).
HardLight	Hard light (Advanced OpenGL blending).
SoftLight	Soft light (Advanced OpenGL blending).
Difference	Difference (Advanced OpenGL blending).
Exclusion	Exclusion (Advanced OpenGL blending).
HSLHue	HSL Hue (Advanced OpenGL blending).
HSLSaturation	HSL saturation (Advanced OpenGL blending).
HSLColor	HSL color (Advanced OpenGL blending).
HSLLuminosity	HSL luminosity (Advanced OpenGL blending).

Description

Built-in temporary render textures produced during camera's rendering.

When camera is rendering the Scene, in some cases it can produce temporary render textures in the process (e.g. depth textures, deferred G-buffer etc.). This enum indicates these temporary render textures.

BuiltinRenderTextureType can be used as a RenderTargetIdentifier in some functions of CommandBuffer.

See Also: CommandBuffer, RenderTargetIdentifier.

Properties

PropertyName	A globally set property name.
BufferPtr	The raw RenderBuffer pointer to be used.
RenderTexture	The given RenderTexture.
CurrentActive	Currently active render target.
CameraTarget	Target texture of currently rendering camera.
Depth	Camera's depth texture.
DepthNormals	Camera's depth+normals texture.
ResolvedDepth	Resolved depth buffer from deferred.
PrepassNormalsSpec	Deferred lighting (normals+specular) G-buffer.
PrepassLight	Deferred lighting light buffer.
PrepassLightSpec	Deferred lighting HDR specular light buffer (Xbox 360 only).
GBuffer0	Deferred shading G-buffer #0 (typically diffuse color).
GBuffer1	Deferred shading G-buffer #1 (typically specular + roughness).
GBuffer2	Deferred shading G-buffer #2 (typically normals).
GBuffer3	Deferred shading G-buffer #3 (typically emission/lighting).
Reflections	Reflections gathered from default reflection and reflections probes.
MotionVectors	Motion Vectors generated when the camera has motion vectors enabled.
GBuffer4	Deferred shading G-buffer #4 (typically occlusion mask for static lights if any).
GBuffer5	G-buffer #5 Available.

GBuffer6	G-buffer #6 Available.
GBuffer7	G-buffer #7 Available.

Description

Defines set by editor when compiling shaders, depending on target platform and tier.

See Also: GraphicsSettings.HasShaderDefine.

Properties

UNITY_NO_DXT5nm	UNITY_NO_DXT5nm is set when compiling shader for platform that do not support DXT5NM, meaning that normal maps will be encoded in RGB instead.
UNITY_NO_RGBM	UNITY_NO_RGBM is set when compiling shader for platform that do not support RGBM, so dLDR will be used instead.
UNITY_ENABLE_REFLECTION_BUFFERS	UNITY_ENABLE_REFLECTION_BUFFERS is set when deferred shading renders reflection probes in deferred mode. With this option set reflections are rendered into a per-pixel buffer. This is similar to the way lights are rendered into a per-pixel buffer. UNITY_ENABLE_REFLECTION_BUFFERS is on by default when using deferred shading, but you can turn it off by setting “No support” for the Deferred Reflections shader option in Graphics Settings. When the setting is off, reflection probes are rendered per-object, similar to the way forward rendering works.
UNITY_FRAMEBUFFER_FETCH_AVAILABLE	UNITY_FRAMEBUFFER_FETCH_AVAILABLE is set when compiling shaders for platforms where framebuffer fetch is potentially available.
UNITY_ENABLE_NATIVE_SHADOW_LOOKUPS	UNITY_ENABLE_NATIVE_SHADOW_LOOKUPS enables use of built-in shadow comparison samplers on OpenGL ES 2.0.
UNITY_METAL_SHADOWS_USE_POINT_FILTERING	UNITY_METAL_SHADOWS_USE_POINT_FILTERING is set if shadow sampler should use point filtering on iOS Metal.
UNITY_NO_SCREENSPACE_SHADOWS	UNITY_NO_SCREENSPACE_SHADOWS is set when screenspace cascaded shadow maps are disabled.
UNITY_USE_DITHER_MASK_FOR_ALPHABLENDED_SHADOWS	UNITY_USE_DITHER_MASK_FOR_ALPHABLENDED_SHADOWS is set when Semitransparent Shadows are enabled.
UNITY_PBS_USE_BRDF1	UNITY_PBS_USE_BRDF1 is set if Standard Shader BRDF1 should be used.
UNITY_PBS_USE_BRDF2	UNITY_PBS_USE_BRDF2 is set if Standard Shader BRDF2 should be used.
UNITY_PBS_USE_BRDF3	UNITY_PBS_USE_BRDF3 is set if Standard Shader BRDF3 should be used.
UNITY_NO_FULL_STANDARD_SHADER	UNITY_NO_FULL_STANDARD_SHADER is set if Standard shader BRDF3 with extra simplifications should be used.

UNITY_SPECCUBE_BOX_PROJECTION	UNITY_SPECCUBE_BLENDING is set if Reflection Probes Box Projection is enabled.
UNITY_SPECCUBE_BLENDING	UNITY_SPECCUBE_BLENDING is set if Reflection Probes Blending is enabled.
UNITY_ENABLE_DETAIL_NORMALMAP	UNITY_ENABLE_DETAIL_NORMALMAP is set if Detail Normal Map should be sampled if assigned.
SHADER_API_MOBILE	SHADER_API_MOBILE is set when compiling shader for mobile platforms.
SHADER_API_DESKTOP	SHADER_API_DESKTOP is set when compiling shader for "desktop" platforms.
UNITY_HARDWARE_TIER1	UNITY_HARDWARE_TIER1 is set when compiling shaders for GraphicsTier.Tier1.
UNITY_HARDWARE_TIER2	UNITY_HARDWARE_TIER2 is set when compiling shaders for GraphicsTier.Tier2.
UNITY_HARDWARE_TIER3	UNITY_HARDWARE_TIER3 is set when compiling shaders for GraphicsTier.Tier3.
UNITY_COLORSPACE_GAMMA	UNITY_COLORSPACE_GAMMA is set when compiling shaders for Gamma Color Space.
UNITY_LIGHT_PROBE_PROXY_VOLUME	UNITY_LIGHT_PROBE_PROXY_VOLUME is set when Light Probe Proxy Volume feature is supported by the current graphics API and is enabled in the current Tier Settings(Graphics Settings).
UNITY_HALF_PRECISION_FRAGMENT_SHADER_REGISTERS	UNITY_HALF_PRECISION_FRAGMENT_SHADER_REGISTERS is set automatically for platforms that don't require full floating-point precision support in fragment shaders.
UNITY_LIGHTMAP_DLDR_ENCODING	UNITY_LIGHTMAP_DLDR_ENCODING is set when lightmap textures are using double LDR encoding to store the values in the texture.
UNITY_LIGHTMAP_RGBM_ENCODING	UNITY_LIGHTMAP_RGBM_ENCODING is set when lightmap textures are using RGBM encoding to store the values in the texture.
UNITY_LIGHTMAP_FULL_HDR	UNITY_LIGHTMAP_FULL_HDR is set when lightmap textures are not using any encoding to store the values in the texture.

Description

Built-in shader modes used by GraphicsSettings.

See Also: GraphicsSettings.SetShaderMode, BuiltinShaderType, Graphics Settings.

Properties

Disabled	Don't use any shader, effectively disabling the functionality.
UseBuiltin	Use built-in shader (default).
UseCustom	Use custom shader instead of built-in one.

Description

Built-in shader types used by GraphicsSettings.

See Also: GraphicsSettings.SetShaderMode, BuiltinShaderMode, Graphics Settings.

Properties

DeferredShading	Shader used for deferred shading calculations.
DeferredReflections	Shader used for deferred reflection probes.
LegacyDeferredLighting	Shader used for legacy deferred lighting calculations.
ScreenSpaceShadows	Shader used for screen-space cascaded shadows.
DepthNormals	Shader used for depth and normals texture when enabled on a Camera.
MotionVectors	Shader used for Motion Vectors when enabled on a Camera.
LightHalo	Default shader used for light halos.
LensFlare	Default shader used for lens flares.

Description

Defines a place in camera's rendering to attach CommandBuffer objects to.

Unity's rendering loop can be extended by adding so called "command buffers" at various points in camera rendering. For example, you could add some custom geometry to be drawn right after the skybox is drawn.

See Also: CommandBuffer, LightEvent, command buffers overview.

Properties

BeforeDepthTexture	Before camera's depth texture is generated.
AfterDepthTexture	After camera's depth texture is generated.
BeforeDepthNormalsTexture	Before camera's depth+normals texture is generated.
AfterDepthNormalsTexture	After camera's depth+normals texture is generated.
BeforeGBuffer	Before deferred rendering G-buffer is rendered.
AfterGBuffer	After deferred rendering G-buffer is rendered.
BeforeLighting	Before lighting pass in deferred rendering.
AfterLighting	After lighting pass in deferred rendering.
BeforeFinalPass	Before final geometry pass in deferred lighting.
AfterFinalPass	After final geometry pass in deferred lighting.
BeforeForwardOpaque	Before opaque objects in forward rendering.
AfterForwardOpaque	After opaque objects in forward rendering.
BeforeImageEffectsOpaque	Before image effects that happen between opaque & transparent objects.
AfterImageEffectsOpaque	After image effects that happen between opaque & transparent objects.
BeforeSkybox	Before skybox is drawn.
AfterSkybox	After skybox is drawn.
BeforeForwardAlpha	Before transparent objects in forward rendering.
AfterForwardAlpha	After transparent objects in forward rendering.
BeforeImageEffects	Before image effects.
AfterImageEffects	After image effects.

AfterEverything	After camera has done rendering everything.
BeforeReflections	Before reflections pass in deferred rendering.
AfterReflections	After reflections pass in deferred rendering.
BeforeHaloAndLensFlares	Before halo and lens flares.
AfterHaloAndLensFlares	After halo and lens flares.

Description

The HDR mode to use for rendering.

When HDR is enabled for the current Graphics Tier this selects the format to use for the HDR buffer.

See Also: TierSettings.hdr, Camera.allowHDR.

Properties

FP16	Uses RenderTextureFormat.ARGBHalf.
R11G11B10	Uses RenderTextureFormat.RGB111110Float.

Description

Specifies which color components will get written into the target framebuffer.

Properties

Alpha	Write alpha component.
Blue	Write blue component.
Green	Write green component.
Red	Write red component.
All	Write all components (R, G, B and Alpha).

Description

Depth or stencil comparison function.

Properties

Disabled	Depth or stencil test is disabled.
Never	Never pass depth or stencil test.
Less	Pass depth or stencil test when new value is less than old one.
Equal	Pass depth or stencil test when values are equal.
LessEqual	Pass depth or stencil test when new value is less or equal than old one.
Greater	Pass depth or stencil test when new value is greater than old one.
NotEqual	Pass depth or stencil test when values are different.
GreaterEqual	Pass depth or stencil test when new value is greater or equal than old one.
Always	Always pass depth or stencil test.

Description

Describes the desired characteristics with respect to prioritisation and load balancing of the queue that a command buffer being submitted via Graphics.ExecuteCommandBufferAsync or [[ScriptableRenderContext.ExecuteCommandBufferAsync] should be sent to.

Properties

Default	This queue type would be the choice for compute tasks supporting or as optimisations to graphics processing. CommandBuffers sent to this queue would be expected to complete within the scope of a single frame and likely be synchronised with the graphics queue via GPUFence's. Dispatches on default queue types would execute at a lower priority than graphics queue tasks.
Background	Background queue types would be the choice for tasks intended to run for an extended period of time, e.g for most of a frame or for several frames. Dispatches on background queues would execute at a lower priority than gfx queue tasks.
Urgent	This queue type would be the choice for compute tasks requiring processing as soon as possible and would be prioritised over the graphics queue.

Description

Support for various Graphics.CopyTexture cases.

Most modern platforms and graphics APIs support quite flexible texture copy (e.g. copy from a RenderTexture into a Cubemap face). However some older systems might not support certain parts of texture copy functionality. This enum indicates support for this. Use SystemInfo.copyTextureSupport to check for support before calling Graphics.CopyTexture.

Direct3D11, DirectD12 and PS4 platforms generally support flexible texture copy (all CopyTextureSupport flags are set).

OpenGL supports flexible texture copy since OpenGL 4.3; OpenGL ES supports flexible texture copy since OpenGL ES 3.1 aep; on earlier versions there's no copy support right now (CopyTextureSupport.None).

Direct3D9 systems have somewhat limited texture copy support (can't copy 3D textures, and can't copy between textures and render textures).

Metal and WebGL currently do not have texture copy support (CopyTextureSupport.None).

See Also: Graphics.CopyTexture, SystemInfo.copyTextureSupport.

Properties

None	No support for Graphics.CopyTexture.
Basic	Basic Graphics.CopyTexture support.
Copy3D	Support for Texture3D in Graphics.CopyTexture.
DifferentTypes	Support for Graphics.CopyTexture between different texture types.
TextureToRT	Support for Texture to RenderTexture copies in Graphics.CopyTexture.
RTToTexture	Support for RenderTexture to Texture copies in Graphics.CopyTexture.

Description

Backface culling mode.

Properties

Off	Disable culling.
Front	Cull front-facing geometry.
Back	Cull back-facing geometry.

Description

Default reflection mode.

Unity can use a custom texture or generate a specular reflection texture from the skybox.

See Also: RenderSettings.defaultReflectionMode, Lighting Window.

Properties

Skybox	Skybox-based default reflection.
Custom	Custom default reflection.

Description

Graphics device API type.

Many different low-level graphics APIs can be used by Unity. If for some reason you need to know whether Direct3D 9 or 11 is being used, or OpenGL ES 2 or 3, you can use SystemInfo.graphicsDeviceType to check for that.

See Also: SystemInfo.graphicsDeviceType.

Properties

Direct3D11	Direct3D 11 graphics API.
Null	No graphics API.
OpenGLES2	OpenGL ES 2.0 graphics API.
OpenGLES3	OpenGL ES 3.0 graphics API.
PlayStation4	PlayStation 4 graphics API.
XboxOne	Xbox One graphics API using Direct3D 11.
Metal	iOS Metal graphics API.
OpenGLCore	OpenGL (Core profile - GL3 or later) graphics API.
Direct3D12	Direct3D 12 graphics API.
Vulkan	Vulkan (EXPERIMENTAL).
Switch	Nintendo Switch graphics API.
XboxOneD3D12	Xbox One graphics API using Direct3D 12.

Description

Graphics Tier. See Also: Graphics.activeTier.

Properties

Tier1	The first graphics tier (Low) - corresponds to shader define UNITY_HARDWARE_TIER1.
Tier2	The second graphics tier (Medium) - corresponds to shader define UNITY_HARDWARE_TIER2.
Tier3	The third graphics tier (High) - corresponds to shader define UNITY_HARDWARE_TIER3.

Description

Format of the mesh index buffer data.

Index buffer can either be 16 bit (supports up to 65535 vertices in a mesh), or 32 bit (supports up to 4 billion vertices). Default index format is 16 bit, since that takes less memory and bandwidth.

Note that GPU support for 32 bit indices is not guaranteed on all platforms; for example Android devices with Mali-400 GPU do not support them. When using 32 bit indices on such a platform, a warning message will be logged and mesh will not render.

See Also: Mesh.indexFormat, ModelImporter.indexFormat.

Properties

UInt16	16 bit mesh index buffer format.
UInt32	32 bit mesh index buffer format.

Description

Defines a place in light's rendering to attach CommandBuffer objects to.

Unity's rendering loop can be extended by adding so called "command buffers" at various points in light rendering; mostly related to shadows. For example, you could do custom processing of the shadow map after it is rendered.

See Also: CommandBuffer, CameraEvent, command buffers overview.

Properties

BeforeShadowMap	Before shadowmap is rendered.
AfterShadowMap	After shadowmap is rendered.
BeforeScreenspaceMask	Before directional light screenspace shadow mask is computed.
AfterScreenspaceMask	After directional light screenspace shadow mask is computed.
BeforeShadowMapPass	Before shadowmap pass is rendered.
AfterShadowMapPass	After shadowmap pass is rendered.

Description

Light probe interpolation type.

See Also: Light Probes, Light Probe Proxy Volume.

Properties

Off	Light Probes are not used. The Scene's ambient probe is provided to the shader.
BlendProbes	Simple light probe interpolation is used.
UseProxyVolume	Uses a 3D grid of interpolated light probes.
CustomProvided	The light probe shader uniform values are extracted from the material property block set on the renderer.

Description

Shadow resolution options for a Light.

See Also: Light.shadowResolution, QualitySettings.shadowResolution, Shadow map size calculation.

Properties

FromQualitySettings	Use resolution from QualitySettings (default).
Low	Low shadow map resolution.
Medium	Medium shadow map resolution.
High	High shadow map resolution.
VeryHigh	Very high shadow map resolution.

Description

Opaque object sorting mode of a Camera.

Opaque objects are sorted by various criteria (sorting layers, shader queues, materials, distance, lightmaps etc.) to maximize both the CPU efficiency (reduce number of state changes and improve draw call batching), and to maximize GPU efficiency (many GPUs prefer rough front-to-back rendering order for faster rejection of invisible surfaces).

By default, opaque objects are grouped in rough front-to-back buckets, on the GPUs where doing that is beneficial. There are GPUs where doing this distance based sorting is not really helpful (most notably, PowerVR/Apple GPUs), and so on these GPUs the distance based sorting is not done by default.

The Camera.opaqueSortMode property lets you override this default behavior. For example, you might want to never do distance-based sorting for opaque objects, if you know you need much more CPU performance than GPU performance.

See Also: Camera.opaqueSortMode.

Properties

Default	Default opaque sorting mode.
FrontToBack	Do rough front-to-back sorting of opaque objects.
NoDistanceSort	Do not sort opaque objects by distance.

Description

Shader pass type for Unity's lighting pipeline.

This corresponds to "LightMode" tag in the shader pass, see Pass tags.

Properties

Normal	Regular shader pass that does not interact with lighting.
Vertex	Legacy vertex-lit shader pass.
VertexLM	Legacy vertex-lit shader pass, with mobile lightmaps.
ForwardBase	Forward rendering base pass.
ForwardAdd	Forward rendering additive pixel light pass.
LightPrePassBase	Legacy deferred lighting (light pre-pass) base pass.
LightPrePassFinal	Legacy deferred lighting (light pre-pass) final pass.
ShadowCaster	Shadow caster & depth texure shader pass.
Deferred	Deferred Shading shader pass.
Meta	Shader pass used to generate the albedo and emissive values used as input to lightmapping.
MotionVectors	Motion vector render pass.
ScriptableRenderPipeline	Custom scriptable pipeline.
ScriptableRenderPipelineDefaultUnlit	Custom scriptable pipeline when lightmode is set to default unlit or no light mode is set.

Description

How much CPU usage to assign to the final lighting calculations at runtime.

How many CPU worker threads to create for Realtime Global Illumination lighting calculations in the Player. Increasing this makes the system react faster to changes in lighting at a cost of using more CPU time. The higher the CPU Usage value, the more worker threads are created for solving Realtime GI.

Please note that some platforms will allow all CPUs to be occupied by worker threads whilst some have a max limit:
Xbox One: 4 CPU cores.
PS4: 4 CPU cores.
Android: If the device is a bigLittle architecture, only the little CPUs will be used, otherwise it is CPUs - 1.

Properties

Low	25% of the allowed CPU threads are used as worker threads.
Medium	50% of the allowed CPU threads are used as worker threads.
High	75% of the allowed CPU threads are used as worker threads.
Unlimited	100% of the allowed CPU threads are used as worker threads.

Description

Determines how Unity will compress baked reflection cubemap.

Properties

Uncompressed	Baked Reflection cubemap will be left uncompressed.
Compressed	Baked Reflection cubemap will be compressed.
Auto	Baked Reflection cubemap will be compressed if compression format is suitable.

Description

Values for ReflectionProbe.clearFlags, determining what to clear when rendering a ReflectionProbe.

See Also: reflection probe component.

Properties

Skybox	Clear with the skybox.
SolidColor	Clear with a background color.

Description

Reflection probe's update mode.

Properties

Baked	Reflection probe is baked in the Editor.
Realtime	Reflection probe is updating in realtime.
Custom	Reflection probe uses a custom texture specified by the user.

Description

An enum describing the way a realtime reflection probe refreshes in the Player.

Properties

OnAwake	Causes the probe to update only on the first frame it becomes visible. The probe will no longer update automatically, however you may subsequently use RenderProbe to refresh the probeSee Also: ReflectionProbe.RenderProbe.
EveryFrame	Causes Unity to update the probe's cubemap every frame. Note that updating a probe is very costly. Setting this option on too many probes could have a significant negative effect on frame rate. Use time-slicing to help improve performance.See Also: ReflectionProbeTimeSlicingMode.
ViaScripting	Sets the probe to never be automatically updated by Unity while your game is running. Use this to completely control the probe refresh behavior by script.See Also: ReflectionProbe.RenderProbe.

Description

When a probe's ReflectionProbe.refreshMode is set to ReflectionProbeRefreshMode.EveryFrame, this enum specify whether or not Unity should update the probe's cubemap over several frames or update the whole cubemap in one frame. Updating a probe's cubemap is a costly operation. Unity needs to render the entire Scene for each face of the cubemap, as well as perform special blurring in order to get glossy reflections. The impact on frame rate can be significant. Time-slicing helps maintaning a more constant frame rate during these updates by performing the rendering over several frames.

Properties

AllFacesAtOnce	Instructs Unity to use time-slicing by first rendering all faces at once, then spreading the remaining work over the next 8 frames. Using this option, updating the probe will take 9 frames.
IndividualFaces	Instructs Unity to spread the rendering of each face over several frames. Using this option, updating the cubemap will take 14 frames. This option greatly reduces the impact on frame rate, however it may produce incorrect results, especially in Scenes where lighting conditions change over these 14 frames.
NoTimeSlicing	Unity will render the probe entirely in one frame.

Description

Reflection Probe usage.

Properties

Off	Reflection probes are disabled, skybox will be used for reflection.
BlendProbes	Reflection probes are enabled. Blending occurs only between probes, useful in indoor environments. The renderer will use default reflection if there are no reflection probes nearby, but no blending between default reflection and probe will occur.
BlendProbesAndSkybox	Reflection probes are enabled. Blending occurs between probes or probes and default reflection, useful for outdoor environments.
Simple	Reflection probes are enabled, but no blending will occur between probes when there are two overlapping volumes.

Description

This enum describes what should be done on the render target when it is activated (loaded).

When the GPU starts rendering into a render target, this setting specifies the action that should be performed on the existing contents of the surface. Tile-based GPUs may get performance advantage if the load action is Clear or DontCare. The user should avoid using RenderBufferLoadAction.Load whenever possible.

Please note that not all platforms have load/store actions, so this setting might be ignored at runtime. Generally mobile-oriented graphics APIs (OpenGL ES, Metal) take advantage of these settings.

Properties

Load	When this RenderBuffer is activated, preserve the existing contents of it. This setting is expensive on tile-based GPUs and should be avoided whenever possible.
Clear	Upon activating the render buffer, clear its contents. Currently only works together with the RenderPass API.
DontCare	When this RenderBuffer is activated, the GPU is instructed not to care about the existing contents of that RenderBuffer. On tile-based GPUs this means that the RenderBuffer contents do not need to be loaded into the tile memory, providing a performance boost.

Description

This enum describes what should be done on the render target when the GPU is done rendering into it.

When the GPU is done rendering into a render target, this setting specifies the action that should be performed on the rendering results. Tile-based GPUs may get performance advantage if the store action is DontCare. For example, this setting can be useful if the depth buffer contents are not needed after rendering the frame.

Please note that not all platforms have load/store actions, so this setting might be ignored at runtime. Generally mobile-oriented graphics APIs (OpenGL ES, Metal) take advantage of these settings.

Properties

Store	The RenderBuffer contents need to be stored to RAM. If the surface has MSAA enabled, this stores the non-resolved surface.
Resolve	Resolve the (MSAA'd) surface. Currently only used with the RenderPass API.
StoreAndResolve	Resolve the (MSAA'd) surface, but also store the multisampled version. Currently only used with the RenderPass API.
DontCare	The contents of the RenderBuffer are not needed and can be discarded. Tile-based GPUs will skip writing out the surface contents altogether, providing performance boost.

Description

Determine in which order objects are renderered.

This way for example transparent objects are rendered after opaque objects, and so on.

See Also: Material.renderQueue, Shader.renderQueue, subshader tags.

Properties

Background	This render queue is rendered before any others.
Geometry	Opaque geometry uses this queue.
AlphaTest	Alpha tested geometry uses this queue.
GeometryLast	Last render queue that is considered "opaque".
Transparent	This render queue is rendered after Geometry and AlphaTest, in back-to-front order.
Overlay	This render queue is meant for overlay effects.

Description

How shadows are cast from this object.

See Also: Renderer.shadowCastingMode.

Properties

Off	No shadows are cast from this object.
On	Shadows are cast from this object.
TwoSided	Shadows are cast from this object, treating it as two-sided.
ShadowsOnly	Object casts shadows, but is otherwise invisible in the Scene.

Description

Allows precise control over which shadow map passes to execute CommandBuffer objects attached using Light.AddCommandBuffer.

These flags only take effect when used with Rendering.LightEvent/BeforeShadowMapPass or Rendering.LightEvent/AfterShadowMapPass.

Properties

PointlightPositiveX	+X point light shadow cubemap face.
PointlightNegativeX	-X point light shadow cubemap face.
PointlightPositiveY	+Y point light shadow cubemap face.
PointlightNegativeY	-Y point light shadow cubemap face.
PointlightPositiveZ	+Z point light shadow cubemap face.
PointlightNegativeZ	-Z point light shadow cubemap face.
DirectionalCascade0	First directional shadow map cascade.
DirectionalCascade1	Second directional shadow map cascade.
DirectionalCascade2	Third directional shadow map cascade.
DirectionalCascade3	Fourth directional shadow map cascade.
Spotlight	Spotlight shadow pass.
Pointlight	All point light shadow passes.
Directional	All directional shadow map passes.
All	All shadow map passes.

Description

Used by CommandBuffer.SetShadowSamplingMode.

Properties

CompareDepths	Default shadow sampling mode: sampling with a comparison filter.
RawDepth	Shadow sampling mode for sampling the depth value.
None	In ShadowSamplingMode.None, depths are not compared. Use this value if a Texture is not a shadowmap.

Description

Specifies the operation that's performed on the stencil buffer when rendering.

Properties

Keep	Keeps the current stencil value.
Zero	Sets the stencil buffer value to zero.
Replace	Replace the stencil buffer value with reference value (specified in the shader).
IncrementSaturate	Increments the current stencil buffer value. Clamps to the maximum representable unsigned value.
DecrementSaturate	Decrements the current stencil buffer value. Clamps to 0.
Invert	Bitwise inverts the current stencil buffer value.
IncrementWrap	Increments the current stencil buffer value. Wraps stencil buffer value to zero when incrementing the maximum representable unsigned value.
DecrementWrap	Decrements the current stencil buffer value. Wraps stencil buffer value to the maximum representable unsigned value when decrementing a stencil buffer value of zero.

Description

Broadly describes the stages of processing a draw call on the GPU.

Used to insert a GPU Fence after or wait for a particular phase of the draw call processing to complete.

Properties

VertexProcessing	All aspects of vertex processing.
PixelProcessing	The process of creating and shading the fragments.

Description

Texture "dimension" (type).

Indicates type of a texture (2D texture, cubemap, 3D volume texture etc.).

See Also: Texture.dimension, MaterialProperty.textureDimension.

Properties

Unknown	Texture type is not initialized or unknown.
None	No texture is assigned.
Any	Any texture type.
Tex2D	2D texture (Texture2D).
Tex3D	3D volume texture (Texture3D).
Cube	Cubemap texture.
Tex2DArray	2D array texture (Texture2DArray).
CubeArray	Cubemap array texture (CubemapArray).

Description

A flag representing each UV channel.

Properties

UV0	First UV channel.
UV1	Second UV channel.
UV2	Third UV channel.
UV3	Fourth UV channel.

Description

Run-time data structure for *.unity file.

Properties

buildIndex	Return the index of the Scene in the Build Settings.
isDirty	Returns true if the Scene is modifed.
isLoaded	Returns true if the Scene is loaded.
name	Returns the name of the Scene.
path	Returns the relative path of the Scene. Like: "Assets/MyScenes/MyScene.unity".
rootCount	The number of root transforms of this Scene.

Public Methods

GetRootGameObjects	Returns all the root game objects in the Scene.
IsValid	Whether this is a valid Scene. A Scene may be invalid if, for example, you tried to open a Scene that does not exist. In this case, the Scene returned from EditorSceneManager.OpenScene would return False for IsValid.

Operators

operator !=	Returns true if the Scenes are different.
operator ==	Returns true if the Scenes are equal.

Description

Scene management at run-time.

Static Properties

sceneCount	The total number of currently loaded Scenes.
sceneCountInBuildSettings	Number of Scenes in Build Settings.

Static Methods

CreateScene	Create an empty new Scene at runtime with the given name.
GetActiveScene	Gets the currently active Scene.
GetSceneAt	Get the Scene at index in the SceneManager's list of loaded Scenes.
GetSceneByBuildIndex	Get a Scene struct from a build index.
GetSceneByName	Searches through the Scenes loaded for a Scene with the given name.
GetSceneByPath	Searches all Scenes loaded for a Scene that has the given asset path.
LoadScene	Loads the Scene by its name or index in Build Settings.
LoadSceneAsync	Loads the Scene asynchronously in the background.
MergeScenes	This will merge the source Scene into the destinationScene.
MoveGameObjectToScene	Move a GameObject from its current Scene to a new Scene.
SetActiveScene	Set the Scene to be active.
UnloadSceneAsync	Destroys all GameObjects associated with the given Scene and removes the Scene from the SceneManager.

Events

activeSceneChanged	Subscribe to this event to get notified when the active Scene has changed.
sceneLoaded	Add a delegate to this to get notifications when a Scene has loaded.
sceneUnloaded	Add a delegate to this to get notifications when a Scene has unloaded.

Description

Scene and Build Settings related utilities.

Static Methods

GetBuildIndexByScenePath	Get the build index from a Scene path.
GetScenePathByBuildIndex	Get the Scene path from a build index.

Description

Used when loading a Scene in a player.

Use LoadSceneMode to choose what type of Scene loads when using SceneManager.LoadScene. The available modes are Single and Additive.

Single mode loads a standard Unity Scene which then appears on its own in the Hierarchy window. Additive loads a Scene which appears in the Hierarchy window while another is active.

```
using UnityEngine;
using UnityEngine.SceneManagement;


public class Example : MonoBehaviour
{
    void OnGUI()
    {
        //This displays a Button on the screen at position (20,30), width 150 an
        if (GUI.Button(new Rect(20, 30, 150, 30), "Other Scene Single"))
        {
            //The SceneManager loads your new Scene as a single Scene (not overl
            SceneManager.LoadScene("YourScene", LoadSceneMode.Single);
        }


        //Whereas pressing this Button loads the Additive Scene.
        if (GUI.Button(new Rect(20, 60, 150, 30), "Other Scene Additive"))
        {
            //SceneManager loads your new Scene as an extra Scene (overlapping t
            SceneManager.LoadScene("YourScene", LoadSceneMode.Additive);
        }
    }
}
```

Properties

Single	Closes all current loaded Scenes and loads a Scene.
Additive	Adds the Scene to the current loaded Scenes.

Description

PreserveAttribute prevents byte code stripping from removing a class, method, field, or property.

When you create a build, Unity will try to strip unused code from your project. This is great to get small builds. However, sometimes you want some code to not be stripped, even if it looks like it is not used. This can happen for instance if you use reflection to call a method, or instantiate an object of a certain class. You can apply the [Preserve] attribute to classes, methods, fields and properties. In addition to using PreserveAttribute, you can also use the traditional method of a link.xml file to tell the linker to not remove things. PreserveAttribute and link.xml work for both the Mono and IL2CPP scripting backends.

```
using UnityEngine;
using System.Collections;
using System.Reflection;
using UnityEngine.Scripting;


public class NewBehaviourScript : MonoBehaviour
{
    void Start()
    {
        ReflectionExample.InvokeBoinkByReflection();
    }
}


public class ReflectionExample
{
    static public void InvokeBoinkByReflection()
    {
        typeof(ReflectionExample).GetMethod("Boink", BindingFlags.NonPublic | Bi
    }


    // No other code directly references the Boink method, so when when strippin
    // it will be removed unless the [Preserve] attribute is applied.
    [Preserve]
    static void Boink()
    {
        Debug.Log("Boink");
    }
}
```

For 3rd party libraries that do not want to take on a dependency on UnityEngine.dll, it is also possible to define their own PreserveAttribute. The code stripper will respect that too, and it will consider any attribute with the exact name "PreserveAtribute" as a reason not to strip the thing it is applied on, regardless of the namespace or assembly of the attribute.

Description

Use this attribute to rename a field without losing its serialized value.

Suppose you have a class like this:

```
using UnityEngine;


public class MyClass : MonoBehaviour
{
    public string myValue;
}
```

And you would like to create a property to encapsulate myValue without losing any data already assigned to instances of MyClass. You can achieve that by using this attribute:

```
using UnityEngine;
using UnityEngine.Serialization;


public class MyClass : MonoBehaviour
{
    [FormerlySerializedAs("myValue")]
    string m_MyValue;
    public string myValue
    {
        get { return m_MyValue; }
        set { m_MyValue = value; }
    }
}
```

Properties

oldName	The name of the field before the rename.

Constructors

FormerlySerializedAsAttribute

Implements interfaces:ISocialPlatform

Description

iOS GameCenter implementation for network services.

An application bundle ID must be registered on iTunes Connect before it can access GameCenter. This ID must be properly set in the iOS player properties in Unity. When debugging you can use the GameCenter sandbox (a text displaying this is shown when logging on). You must log on in the application to get into sandbox mode, logging on in the GameCenter application will always use the production version.

When using the GameCenterPlatform class in C# you need to include the UnityEngine.SocialPlatforms.GameCenter namespace.

Some things to be aware of when using the generic API:

Authenticate()
If the user is not logged in, a standard GameKit UI is shown where they can log on or create a new user. It is recommended this is done as early as possible.

Achievement descriptions and Leaderboards
The achivements descriptions and leaderboard configurations can be configured in the iTunes Connect portal. Achievements get unique identifiers and the leaderboards use category names as identifiers.

GameCenter Sandbox
Development applications use the GameCenter Sandbox. This is a seperate GameCenter than the live one, nothing is shared between them. It is recommended that you create a seperate user for testing with the GameCenter Sandbox, you should not use your real Apple ID for this. You can only log on to the sandbox through a development application, make sure you are not logged into GameCenter using the GameCenter app before testing begins. You should see *** *Sandbox* *** in the login dialog, if you don't see this then you are logging on to the real one. Sometimes it happens that the OS forgets that the application is using the sandbox and you will be logged on to the real one. If the application has not been submitted to Apple already then this will probably result in an error. To fix this all that needs to be done is to delete the app and redeploy with Xcode. To make another Apple ID a friend of a sandbox user it needs to be a sandbox user as well.

If you start getting errors when accessing GameCenter stating that the application is not recognized you'll need to delete the application completely and re-deploy. Make sure you are not logged on when starting the newly installed application again.

Public Methods

ShowLeaderboardUI	Show the leaderboard UI with a specific leaderboard shown initially with a specific time scope selected.

Static Methods

ResetAllAchievements	Reset all the achievements for the local user.
ShowDefaultAchievementCompletionBanner	Show the default iOS banner when achievements are completed.

Description

The score range a leaderboard query should include.

Properties

count	The total amount of scores retreived.
from	The rank of the first score which is returned.

Constructors

Range	Constructor for a score range, the range starts from a specific value and contains a maxium score count.

Description

Information for a user's achievement.

This defines the relation between a particular achievement (described by IAchievementDescription) and the local user, what progress they have, last date they reported progress and so on.

Use Social.CreateAchievement to create an instance of this object.

Properties

completed	Set to true when percentCompleted is 100.0.
hidden	This achievement is currently hidden from the user.
id	The unique identifier of this achievement.
lastReportedDate	Set by server when percentCompleted is updated.
percentCompleted	Progress for this achievement.

Public Methods

ReportProgress	Send notification about progress on this achievement.

Description

Static data describing an achievement.

Retreive the achievement descriptions by using Social.LoadAchievementDescriptions. GameCenter: These are stored server side and should be fetched when the user has authenticated. These are set in the iTunes Connect.

Properties

achievedDescription	Description when the achivement is completed.
hidden	Hidden achievement are not shown in the list until the percentCompleted has been touched (even if it's 0.0).
id	Unique identifier for this achievement description.
image	Image representation of the achievement.
points	Point value of this achievement.
title	Human readable title.
unachievedDescription	Description when the achivement has not been completed.

Description

The leaderboard contains the scores of all players for a particular game.

Each game can have multiple leaderboards with different scores. A leaderboard object can be customized to perform a particular query. The leaderboard ID defines which leaderboard is being queried and there are filters to narrow down the results, UserScope, TimeScope, Range and SetUserFilter. Use Social.CreateLeaderboard to create an instance of this object.

Properties

id	Unique identifier for this leaderboard.
loading	The leaderboad is in the process of loading scores.
localUserScore	The leaderboard score of the logged in user.
maxRange	The total amount of scores the leaderboard contains.
range	The rank range this leaderboard returns.
scores	The leaderboard scores returned by a query.
timeScope	The time period/scope searched by this leaderboard.
title	The human readable title of this leaderboard.
userScope	The users scope searched by this leaderboard.

Public Methods

LoadScores	Load scores according to the filters set on this leaderboard.
SetUserFilter	Only search for these user IDs.

Implements interfaces:IUserProfile

Description

Represents the local or currently logged in user.

Properties

authenticated	Checks if the current user has been authenticated.
friends	The users friends list.
underage	Is the user underage?

Public Methods

Authenticate	Authenticate the local user to the current active Social API implementation and fetch his profile data.
LoadFriends	Fetches the friends list of the logged in user. The friends list on the Social.localUser instance is populated if this call succeeds.

Description

A game score.

It can be received from a ILeaderboard instance or using the Social.LoadScores call which uses the default leaderboard filters.

Properties

date	The date the score was achieved.
formattedValue	The correctly formatted value of the score, like X points or X kills.
leaderboardID	The ID of the leaderboard this score belongs to.
rank	The rank or position of the score in the leaderboard.
userID	The user who owns this score.
value	The score value achieved.

Public Methods

ReportScore	Report this score instance.

Description

The generic Social API interface which implementations must inherit.

See Social for details on usage.

Properties

localUser	See Social.localUser.

Public Methods

CreateAchievement	See Social.CreateAchievement.
CreateLeaderboard	See Social.CreateLeaderboard.
LoadAchievementDescriptions	See Social.LoadAchievementDescriptions.
LoadAchievements	See Social.LoadAchievements.
LoadScores	See Social.LoadScores.
LoadUsers	See Social.LoadUsers.
ReportProgress	See Social.ReportProgress.
ReportScore	See Social.ReportScore.
ShowAchievementsUI	See Social.ShowAchievementsUI.
ShowLeaderboardUI	See Social.ShowLeaderboardUI.

Description

Represents generic user instances, like friends of the local user.

Properties

id	This users unique identifier.
image	Avatar image of the user.
isFriend	Is this user a friend of the current logged in user?
state	Presence state of the user.
userName	This user's username or alias.

Description

The scope of time searched through when querying the leaderboard.

Description

The scope of the users searched through when querying the leaderboard.

Description

User presence state.

Properties

Online	The user is online.
OnlineAndAway	The user is online but away from their computer.
OnlineAndBusy	The user is online but set their status to busy.
Offline	The user is offline.
Playing	The user is playing a game.

Description

The PoseDataSource class acts as a container for the GetDatafromSource method call that should be used by PoseProviders wanting to query data for a particular pose.

Static Methods

TryGetDataFromSource	The GetDatafromSource method is used to query data from the XRNode subsystem based on the provided pose source.

Description

The TrackedPoseDriver component applies the current Pose value of a tracked device to the transform of the GameObject.

TrackedPoseDriver can track multiple types of devices including XR HMDs, controllers, and remotes.

Properties

deviceType	The device being tracked by the tracked pose driver
poseProviderComponent	Optional: This field holds the reference to the PoseProvider instance that, if set, will be used to override the behavior of the TrackedPoseDriver. When this field is empty, the TrackedPoseDriver will operate as per usual, with pose data being retrieved from the device or pose settings of the TrackedPoseDriver. When this field is set, the pose data will be provided by the attached PoseProvider. The device or pose fields will be hidden as they are no longer used to control the parent GameObject Transform.
poseSource	The pose being tracked by the tracked pose driver
trackingType	The tracking type being used by the tracked pose driver
updateType	The update type being used by the tracked pose driver
UseRelativeTransform	This is used to indicate whether the TrackedPoseDriver will use the object's original transform as its basis.

Public Methods

SetPoseSource	This method is used to set the device / pose pair for the SpatialTracking.TrackedPoseDriver. setting an invalid combination of these values will return false.

Inherited Members

Properties

enabled	Enabled Behaviours are Updated, disabled Behaviours are not.
isActiveAndEnabled	Has the Behaviour had active and enabled called?
gameObject	The game object this component is attached to. A component is always attached to a game object.

tag	The tag of this game object.
transform	The Transform attached to this GameObject.
runInEditMode	Allow a specific instance of a MonoBehaviour to run in edit mode (only available in the editor).
useGUILayout	Disabling this lets you skip the GUI layout phase.
hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.

Public Methods

BroadcastMessage	Calls the method named methodName on every MonoBehaviour in this game object or any of its children.
CompareTag	Is this game object tagged with tag ?
GetComponent	Returns the component of Type type if the game object has one attached, null if it doesn't.
GetComponentInChildren	Returns the component of Type type in the GameObject or any of its children using depth first search.
GetComponentInParent	Returns the component of Type type in the GameObject or any of its parents.
GetComponents	Returns all components of Type type in the GameObject.
GetComponentsInChildren	Returns all components of Type type in the GameObject or any of its children.
GetComponentsInParent	Returns all components of Type type in the GameObject or any of its parents.
SendMessage	Calls the method named methodName on every MonoBehaviour in this game object.
SendMessageUpwards	Calls the method named methodName on every MonoBehaviour in this game object and on every ancestor of the behaviour.
CancelInvoke	Cancels all Invoke calls on this MonoBehaviour.
Invoke	Invokes the method methodName in time seconds.
InvokeRepeating	Invokes the method methodName in time seconds, then repeatedly every repeatRate seconds.
IsInvoking	Is any invoke on methodName pending?
StartCoroutine	Starts a coroutine.
StopAllCoroutines	Stops all coroutines running on this behaviour.

StopCoroutine	Stops the first coroutine named methodName, or the coroutine stored in routine running on this behaviour.
GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.

Static Methods

print	Logs message to the Unity Console (identical to Debug.Log).
Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Messages

Awake	Awake is called when the script instance is being loaded.
FixedUpdate	This function is called every fixed framerate frame, if the MonoBehaviour is enabled.
LateUpdate	LateUpdate is called every frame, if the Behaviour is enabled.
OnAnimatorIK	Callback for setting up animation IK (inverse kinematics).
OnAnimatorMove	Callback for processing animation movements for modifying root motion.
OnApplicationFocus	Sent to all GameObjects when the player gets or loses focus.
OnApplicationPause	Sent to all GameObjects when the application pauses.

OnApplicationQuit	Sent to all game objects before the application quits.
OnAudioFilterRead	If OnAudioFilterRead is implemented, Unity will insert a custom filter into the audio DSP chain.
OnBecameInvisible	OnBecameInvisible is called when the renderer is no longer visible by any camera.
OnBecameVisible	OnBecameVisible is called when the renderer became visible by any camera.
OnCollisionEnter	OnCollisionEnter is called when this collider/rigidbody has begun touching another rigidbody/collider.
OnCollisionEnter2D	Sent when an incoming collider makes contact with this object's collider (2D physics only).
OnCollisionExit	OnCollisionExit is called when this collider/rigidbody has stopped touching another rigidbody/collider.
OnCollisionExit2D	Sent when a collider on another object stops touching this object's collider (2D physics only).
OnCollisionStay	:ref::OnCollisionStay is called once per frame for every collider/rigidbody that is touching rigidbody/collider.
OnCollisionStay2D	Sent each frame where a collider on another object is touching this object's collider (2D physics only).
OnConnectedToServer	Called on the client when you have successfully connected to a server.
OnControllerColliderHit	OnControllerColliderHit is called when the controller hits a collider while performing a Move.
OnDestroy	Destroying the attached Behaviour will result in the game or Scene receiving OnDestroy.
OnDisable	This function is called when the behaviour becomes disabled.
OnDisconnectedFromServer	Called on the client when the connection was lost or you disconnected from the server.
OnDrawGizmos	Implement OnDrawGizmos if you want to draw gizmos that are also pickable and always drawn.
OnDrawGizmosSelected	Implement OnDrawGizmosSelected to draw a gizmo if the object is selected.
OnEnable	This function is called when the object becomes enabled and active.
OnFailedToConnect	Called on the client when a connection attempt fails for some reason.
OnFailedToConnectToMasterServer	Called on clients or servers when there is a problem connecting to the

	MasterServer.
OnGUI	OnGUI is called for rendering and handling GUI events.
OnJointBreak	Called when a joint attached to the same game object broke.
OnJointBreak2D	Called when a Joint2D attached to the same game object breaks.
OnMasterServerEvent	Called on clients or servers when reporting events from the MasterServer.
OnMouseDown	OnMouseDown is called when the user has pressed the mouse button while over the GUIElement or Collider.
OnMouseDrag	OnMouseDrag is called when the user has clicked on a GUIElement or Collider and is still holding down the mouse.
OnMouseEnter	Called when the mouse enters the GUIElement or Collider.
OnMouseExit	Called when the mouse is not any longer over the GUIElement or Collider.
OnMouseOver	Called every frame while the mouse is over the GUIElement or Collider.
OnMouseUp	OnMouseUp is called when the user has released the mouse button.
OnMouseUpAsButton	OnMouseUpAsButton is only called when the mouse is released over the same GUIElement or Collider as it was pressed.
OnNetworkInstantiate	Called on objects which have been network instantiated with Network.Instantiate.
OnParticleCollision	OnParticleCollision is called when a particle hits a Collider.
OnParticleSystemStopped	OnParticleSystemStopped is called when all particles in the system have died, and no new particles will be born. New particles cease to be created either after Stop is called, or when the duration property of a non-looping system has been exceeded.
OnParticleTrigger	OnParticleTrigger is called when any particles in a particle system meet the conditions in the trigger module.
OnPlayerConnected	Called on the server whenever a new player has successfully connected.
OnPlayerDisconnected	Called on the server whenever a player disconnected from the server.
OnPostRender	OnPostRender is called after a camera finished rendering the Scene.
OnPreCull	OnPreCull is called before a camera culls the Scene.
OnPreRender	OnPreRender is called before a camera starts rendering the Scene.
OnRenderImage	OnRenderImage is called after all rendering is complete to render

	image.
OnRenderObject	OnRenderObject is called after camera has rendered the Scene.
OnSerializeNetworkView	Used to customize synchronization of variables in a script watched by a network view.
OnServerInitialized	Called on the server whenever a Network.InitializeServer was invoked and has completed.
OnTransformChildrenChanged	This function is called when the list of children of the transform of the GameObject has changed.
OnTransformParentChanged	This function is called when the parent property of the transform of the GameObject has changed.
OnTriggerEnter	OnTriggerEnter is called when the Collider other enters the trigger.
OnTriggerEnter2D	Sent when another object enters a trigger collider attached to this object (2D physics only).
OnTriggerExit	OnTriggerExit is called when the Collider other has stopped touching the trigger.
OnTriggerExit2D	Sent when another object leaves a trigger collider attached to this object (2D physics only).
OnTriggerStay	OnTriggerStay is called once per physics update for every Collider other that is touching the trigger.
OnTriggerStay2D	Sent each frame where another object is within a trigger collider attached to this object (2D physics only).
OnValidate	This function is called when the script is loaded or a value is changed in the inspector (Called in the editor only).
OnWillRenderObject	OnWillRenderObject is called for each camera if the object is visible and not a UI element.
Reset	Reset to default values.
Start	Start is called on the frame when a script is enabled just before any of the Update methods are called the first time.
Update	Update is called every frame, if the MonoBehaviour is enabled.

Description

This is used to indicate which pose the TrackedPoseDriver is currently tracking.

Properties

GenericXRDevice	An Generic XR Devices, use this value for HMD and AR Mobile device tracking
GenericXRController	An XR Controller, use this value for controllers
GenericXRRemote	An XR Remote, use this value for mobile remotes

Description

This is used to indicate which pose the Tracked Pose Driver is currently tracking

Properties

LeftEye	The left eye of a HMD style device
RightEye	The right eye of a HMD style device
Center	The center eye of a HMD style device, this is usually the default for most HMDs
Head	The head eye of a HMD style device
LeftPose	The left hand controller pose
RightPose	The right hand contoller pose
ColorCamera	The color camera of a mobile device

Description

This enum is used to indiciate which parts of the pose will be applied to the parent transform

Properties

RotationAndPosition	With this setting, both the pose's rotation and position will be applied to the parent transform
RotationOnly	With this setting, only the pose's rotation will be applied to the parent transform
PositionOnly	With this setting, only the pose's position will be applied to the parent transform

Description

This is used to indicate what points in the frame the tracked pose driver will sample input

Properties

UpdateAndBeforeRender	Sample input at both update, and directly before rendering. For smooth head pose tracking, we recommend using this value as it will provide the lowest input latency for the device. This is the default value for the UpdateType option
Update	Only sample input during the update phase of the frame.
BeforeRender	Only sample input directly before rendering

Description

Helper utilities for accessing Sprite data.

Static Methods

GetInnerUV	Inner UV's of the Sprite.
GetMinSize	Minimum width and height of the Sprite.
GetOuterUV	Outer UV's of the Sprite.
GetPadding	Return the padding on the sprite.

Description

Compares two Color objects for equality within the given tolerance.

Use this class to compare two Color objects for equality, with `NUnit` constraints. It implements the System.Collections.Generic.IEqualityComparer<Color> interface. Use the ColorEqualityComparer.Instance comparer to use the default error *0.01f*. For any other error, use the one argument constructor to create a comparer.

It uses Utils.AreFloatsEqualAbsoluteError to compare the `r, g, b, a` attributes of Color.

Static Properties

Instance	A comparer instance with default error 0.01f.

Constructors

ColorEqualityComparer	Creates an instance with a custom error value.

Public Methods

Equals	Compares the actual and expected Color objects for equality.
GetHashCode	Hash code implementation which returns same hash code for all equal Color objects.

Description

Float value comparer to work with **NUnit** constraints.

Use this class to compare two *float* values for equality, with `NUnit` constraints. It implements the System.Collections.Generic.IEqualityComparer<float> interface. Use FloatEqualityComparer.Instance comparer to use default error *0.0001f*. For any other error, use the one argument constructor to create a comparer.

It uses Utils.AreFloatsEqual to compare the float values, which is a *relative epsilon comparison*.

Static Properties

Instance	A comparer instance with default error 0.0001f.

Constructors

FloatEqualityComparer	Create an instance with a custom error value.

Public Methods

Equals	Compare the actual and expected float values for equality.
GetHashCode	Hash code implementation which returns same hash code for all equal float values.

Description

Utility to compare two Quaternion objects for equality with NUnit assertion constraints.

It implements the System.Collections.Generic.IEqualityComparer<Quaternion> interface. Use the static instance QuaternionEqualityComparer.Instance to compare with default error *0.00001f*. For any custom error value, use the one argument constructor.

Static Properties

Instance	A comparer instance with default error 0.00001f.

Constructors

QuaternionEqualityComparer	Creates an instance with a custom error value.

Public Methods

Equals	Compares the actual and expected Quaternion objects for equality.
GetHashCode	Hash code implementation which returns same hash code for all equal Quaternion objects.

Description

Contains utility functions like comparing float values.

Static Methods

AreFloatsEqual	Relative epsilon comparison of two float values for equality.
AreFloatsEqualAbsoluteError	Compares two float values for equality.
CreatePrimitive	Creates a GameObject with a primitive mesh renderer.

Description

Compares two Vector2 objects for equality using overloaded == operator.

Use this class to compare two Vector2 objects for equality, with `NUnit` constraints. It implements the System.Collections.Generic.IEqualityComparer<Vector2> interface. It uses the overloaded *operator*== of Vector2 to compare two Vector2 objects.

Static Properties

Instance	A comparer instance to be used with NUnit constrants.

Public Methods

Equals	Compares the actual and expected Vector2 objects for equality.
GetHashCode	Hash code implementation which returns same hash code for all equal Vector2 objects.

Description

Compares two Vector2 objects for equality within given tolerance.

Use this class to compare two Vector2 objects for equality, with `NUnit` constraints. It implements the System.Collections.Generic.IEqualityComparer<Vector2> interface. Use Vector2EqualityComparer.Instance comparer to use default error *0.0001f*. For any other error, use the one argument constructor to create a comparer.

It uses Utils.AreFloatsEqual to compare the `x and y` attributes of Vector2.

Static Properties

Instance	A comparer instance with default error 0.0001f.

Constructors

Vector2EqualityComparer	Creates an instance with a custom error value.

Public Methods

Equals	Compares the actual and expected Vector2 objects for equality.
GetHashCode	Hash code implementation which returns same hash code for all equal Vector2 objects.

Description

Compare two Vector3 objects for equality using overloaded == operator.

Use this class to compare two Vector3 objects for equality, with `NUnit` constraints. It implements the System.Collections.Generic.IEqualityComparer<Vector3> interface. It uses the overloaded *operator*== of Vector3 to compare two Vector3 objects.

Static Properties

Instance	A comparer instance to be used with NUnit constrants.

Public Methods

Equals	Compare the actual and expected Vector3 objects for equality.
GetHashCode	Hash code implementation which returns same hash code for all equal Vector3 objects.

Description

Compares two Vector3 objects for equality within given tolerance.

Use this class to compare two Vector3 objects for equality, with `NUnit` constraints. It implements the System.Collections.Generic.IEqualityComparer<Vector3> interface. Use Vector3EqualityComparer.Instance comparer to use default error *0.0001f*. For any other error, use the one argument constructor to create a comparer.

It uses Utils.AreFloatsEqual to compare the `x, y and z` attributes of Vector3.

Static Properties

Instance	A comparer instance with default error 0.0001f.

Constructors

Vector3EqualityComparer	Creates an instance with a custom error value.

Public Methods

Equals	Compares the actual and expected Vector3 objects for equality.
GetHashCode	Hash code implementation which returns same hash code for all equal Vector3 objects.

Description

Compares two Vector4 objects for equality using overloaded == operator.

Use this class to compare two Vector4 objects for equality, with `NUnit` constraints. It implements the System.Collections.Generic.IEqualityComparer<Vector4> interface. It uses the overloaded *operator*== of Vector4 to compare two Vector4 objects.

Static Properties

Instance	A comparer instance to be used with NUnit constrants.

Public Methods

Equals	Compares the actual and expected Vector4 objects for equality.
GetHashCode	Hash code implementation which returns same hash code for all equal Vector4 objects.

Description

Compares two Vector4 objects for equality within given tolerance.

Use this class to compare two Vector4 objects for equality, with `NUnit` constraints. It implements the System.Collections.Generic.IEqualityComparer<Vector4> interface. Use Vector4EqualityComparer.Instance comparer to use default error *0.0001f*. For any other error, use the one argument constructor to create a comparer.

It uses Utils.AreFloatsEqual to compare the `x, y, z and w` attributes of Vector4.

Static Properties

Instance	A comparer instance with default error 0.0001f.

Constructors

Vector4EqualityComparer	Creates an instance with a custom error value.

Public Methods

Equals	Compares the actual and expected Vector4 objects for equality.
GetHashCode	Hash code implementation which returns same hash code for all equal Vector4 objects.

Description

LogAssert allows you to expect Unity log messages that would normally cause the test to fail.

Static Properties

ignoreFailingMessages	Set this property to true to prevent unexpected error log messages from triggering an assertion. This property is set to false by default.

Static Methods

Expect	Expect a log message of a specfic type. If an error, assertion or exception log is expected, the test will not fail. The test will fail if a log message is expected but does not appear.
NoUnexpectedReceived	Triggers an assert if any logs have been received that were not expected. Returns without asserting if all logs received so far have been registered as expected.

Description

Wrapper for running tests that are imlpemented as MonoBehaviours.

Properties

component	The test component attached this object.
gameObject	The GameObject that hold test component.

Inherited Members

Properties

keepWaiting	Indicates if coroutine should be kept suspended.

Description

A MonoBehaviour test needs to implement this interface.

Properties

IsTestFinished	Indicates when the test is considered finished.

Description

Interface for the method that implements the postbuild step.

Public Methods

Cleanup	Cleanup method that is automatically called after all the test have run.

Description

Interface for the method that implements the prebuild step.

Public Methods

Setup	Setup method that is automatically called before the test run.

Description

Platforms the tests can run on.

Description

Allows you to define a cleanup method for the test that is invoked after all tests are run.

Constructors

PostBuildCleanupAttribute	Points to a class that implements TestTools.IPostbuildCleanup. The method from TestTools.IPostbuildCleanup is executed after all tests are run.

Description

Allows to define a setup method for the test that will be invoked before the test run is started.

Constructors

PrebuildSetupAttribute	Points to a class that implements IPrebuildSetup. The method from IPrebuildSetup is executed before the test run is initiated.

Description

For specifying platforms to run on.

Implement this class to specify which platforms should be tested. Use the RuntimePlatform enumeration to specify the platforms, along with the `include` or `exclude` properties of lists to set whether a test should apply to the platforms. You can also specify which platforms should be tested by passing one or more RuntimePlatform values as parameters to the PlatformAttribute constructor. All platforms are tested if nothing is specified using `RuntimePlatform`, `include` or `exclude`. No tests are excluded if the `include` property is not null. See Also: RuntimePlatform , include , exclude .

```
using UnityEngine;
using UnityEngine.TestTools;
using NUnit.Framework;


[TestFixture]
public class TestClass
{
    [Test]
    [UnityPlatform(RuntimePlatform.WindowsPlayer)]
    public void TestMethod()
    {
        Assert.AreEqual(Application.platform, RuntimePlatform.WindowsPlayer);
    }
}
```

Properties

exclude	Excluded platforms.
include	Included platforms.

Description

Special type of a SetUp that allows to yield before the test is running.

Description

Special type of a TearDown that allows to yield after the test is running.

Description

Special type of a unit test that allows to yield from test in order to skip frames when the test is running.

Description

Class passed onto tiles when information is queried from the tiles.

This handles editor preview tiles when painting on a Tilemap in Editor mode.

Properties

cellBounds	Returns the boundaries of the Tilemap in cell size.
localBounds	Returns the boundaries of the Tilemap in local space size.
origin	The origin of the Tilemap in cell position.
size	The size of the Tilemap in cells.

Public Methods

GetColor	Gets the color of a tile given the XYZ coordinates of a cell in the tile map.
GetComponent	Returns the component of type T if the GameObject of the tile map has one attached, null if it doesn't.
GetSprite	Gets the Sprite used in a tile given the XYZ coordinates of a cell in the tile map.
GetTile	Gets the tile at the given XYZ coordinates of a cell in the tile map.
GetTileFlags	Gets the TileFlags of the Tile at the given position.
GetTransformMatrix	Gets the transform matrix of a tile given the XYZ coordinates of a cell in the tile map.
RefreshTile	Refreshes a tile at the given XYZ coordinates of a cell in the tile map.

Description

Class for a default tile in the Tilemap.

This inherits from TileBase and represents a default tile to be placed in a Tilemap. It implements TileBase.GetTileData for simple rendering of a Sprite in the tile map.

Properties

color	Color of the Tile.
flags	TileFlags of the Tile.
gameObject	GameObject of the Tile.
sprite	Sprite to be rendered at the Tile.
transform	Transform matrix of the Tile.

Public Methods

GetTileData	Retrieves the tile rendering data for the Tile.

Inherited Members

Properties

hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.

Public Methods

GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.
GetTileAnimationData	Retrieves any tile animation data from the scripted tile.
RefreshTile	This method is called when the tile is refreshed.
StartUp	StartUp is called on the first frame of the running Scene.

Static Methods

Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.
CreateInstance	Creates an instance of a scriptable object.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Messages

Awake	This function is called when the ScriptableObject script is started.
OnDestroy	This function is called when the scriptable object will be destroyed.
OnDisable	This function is called when the scriptable object goes out of scope.
OnEnable	This function is called when the object is loaded.

Description

Enum for determining what collider shape is generated for this Tile by the TilemapCollider2D.

Properties

None	No collider shape is generated for the Tile by the TilemapCollider2D.
Sprite	The Sprite outline is used as the collider shape for the Tile by the TilemapCollider2D.
Grid	The grid layout boundary outline is used as the collider shape for the Tile by the TilemapCollider2D.

Description

A Struct for the required data for animating a Tile.

Properties

animatedSprites	The array of sprites that are ordered by appearance in the animation.
animationSpeed	The animation speed.
animationStartTime	The start time of the animation. The animation will begin at this time offset.

Description

Base class for a tile in the Tilemap.

Inherit from this to implement your custom tile to be placed in a Tilemap component.

Public Methods

GetTileAnimationData	Retrieves any tile animation data from the scripted tile.
GetTileData	Retrieves any tile rendering data from the scripted tile.
RefreshTile	This method is called when the tile is refreshed.
StartUp	StartUp is called on the first frame of the running Scene.

Inherited Members

Properties

hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.

Public Methods

GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.

Static Methods

Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.
CreateInstance	Creates an instance of a scriptable object.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Messages

Awake	This function is called when the ScriptableObject script is started.
OnDestroy	This function is called when the scriptable object will be destroyed.
OnDisable	This function is called when the scriptable object goes out of scope.
OnEnable	This function is called when the object is loaded.

Description

A Struct for the required data for rendering a Tile.

Properties

color	Color of the Tile.
flags	TileFlags of the Tile.
gameObject	GameObject of the Tile.
sprite	Sprite to be rendered at the Tile.
transform	Transform matrix of the Tile.

Description

The tile map stores sprites in a layout marked by a Grid component.

Properties

animationFrameRate	The frame rate for all tile animations in the tile map.
cellBounds	Returns the boundaries of the Tilemap in cell size.
color	The color of the tile map layer.
editorPreviewOrigin	The origin of the Tilemap in cell position inclusive of editor preview tiles.
editorPreviewSize	The size of the Tilemap in cells inclusive of editor preview tiles.
layoutGrid	Gets the Grid associated with this tile map.
localBounds	Returns the boundaries of the Tilemap in local space size.
orientation	Orientation of the tiles in the Tilemap.
orientationMatrix	Orientation Matrix of the orientation of the tiles in the Tilemap.
origin	The origin of the Tilemap in cell position.
size	The size of the Tilemap in cells.
tileAnchor	Gets the anchor point of tiles in the Tilemap.

Public Methods

AddTileFlags	Adds the TileFlags onto the Tile at the given position.
BoxFill	Does a box fill with the given tile on the tile map. Starts from given coordinates and fills the limits from start to end (inclusive).
ClearAllEditorPreviewTiles	Clears all editor preview tiles that are placed in the Tilemap.
ClearAllTiles	Clears all tiles that are placed in the Tilemap.
CompressBounds	Compresses the origin and size of the Tilemap to bounds where tiles exist.
ContainsTile	Returns true if the Tilemap contains the given tile. Returns false if not.
EditorPreviewBoxFill	Does an editor preview of a box fill with the given tile on the tile map. Starts from given coordinates and fills the limits from start to end

	(inclusive).
EditorPreviewFloodFill	Does an editor preview of a flood fill with the given tile to place. on the tile map starting from the given coordinates.
FloodFill	Does a flood fill with the given tile to place. on the tile map starting from the given coordinates.
GetCellCenterLocal	Get the logical center coordinate of a grid cell in local space.
GetCellCenterWorld	Get the logical center coordinate of a grid cell in world space.
GetColliderType	Gets the collider type of a tile given the XYZ coordinates of a cell in the tile map.
GetColor	Gets the color of a tile given the XYZ coordinates of a cell in the tile map.
GetEditorPreviewColor	Gets the Color of an editor preview tile given the XYZ coordinates of a cell in the tile map.
GetEditorPreviewSprite	Gets the Sprite used in an editor preview tile given the XYZ coordinates of a cell in the tile map.
GetEditorPreviewTile	Gets the editor preview tile at the given XYZ coordinates of a cell in the tile map.
GetEditorPreviewTileFlags	Gets the TileFlags of the editor preview Tile at the given position.
GetEditorPreviewTransformMatrix	Gets the transform matrix of an editor preview tile given the XYZ coordinates of a cell in the tile map.
GetInstantiatedObject	Gets the GameObject instantiated by a tile given the XYZ coordinates of a cell in the tile map.
GetSprite	Gets the Sprite used in a tile given the XYZ coordinates of a cell in the tile map.
GetTile	Gets the tile at the given XYZ coordinates of a cell in the tile map.
GetTileFlags	Gets the TileFlags of the Tile at the given position.
GetTilesBlock	Retrieves an array of tiles with the given bounds.
GetTransformMatrix	Gets the transform matrix of a tile given the XYZ coordinates of a cell in the tile map.
GetUsedTilesCount	Get the total number of different tiles used in the Tilemap.
GetUsedTilesNonAlloc	Fills the given array with the total number of different tiles used in the Tilemap and returns the number of tiles filled.
HasEditorPreviewTile	Returns whether there is an editor preview tile at the position.

HasTile	Returns whether there is a tile at the position.
RefreshAllTiles	Refreshes all tiles in the tile map. The tile map will retrieve the rendering data, animation data and other data for all tiles and update all relevant components.
RefreshTile	Refreshes a tile at the given coordinates of a cell in the tile map.
RemoveTileFlags	Removes the TileFlags onto the Tile at the given position.
ResizeBounds	Resizes tiles in the Tilemap to bounds defined by origin and size.
SetColliderType	Sets the collider type of a tile given the XYZ coordinates of a cell in the tile map.
SetColor	Sets the color of a tile given the XYZ coordinates of a cell in the tile map.
SetEditorPreviewColor	Sets the color of an editor preview tile given the XYZ coordinates of a cell in the tile map.
SetEditorPreviewTile	Sets an editor preview tile given the XYZ coordinates of a cell in the tile map.
SetEditorPreviewTransformMatrix	Sets the transform matrix of an editor preview tile given the XYZ coordinates of a cell in the tile map.
SetTile	Sets a tile at the given XYZ coordinates of a cell in the tile map.
SetTileFlags	Sets the TileFlags onto the Tile at the given position.
SetTiles	Sets an array of tiles at the given XYZ coordinates of the corresponding cells in the tile map.
SetTilesBlock	Fills bounds with array of tiles.
SetTransformMatrix	Sets the transform matrix of a tile given the XYZ coordinates of a cell in the tile map.
SwapTile	Swaps all existing tiles of changeTile to newTile and refreshes all the swapped tiles.

Inherited Members

Properties

enabled	Enabled Behaviours are Updated, disabled Behaviours are not.
isActiveAndEnabled	Has the Behaviour had active and enabled called?
gameObject	The game object this component is attached to. A component is always attached to a game object.

tag	The tag of this game object.
transform	The Transform attached to this GameObject.
cellGap	The size of the gap between each cell in the layout.
cellLayout	The layout of the cells.
cellSize	The size of each cell in the layout.
cellSwizzle	The cell swizzle for the layout.
hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.

Public Methods

BroadcastMessage	Calls the method named methodName on every MonoBehaviour in this game object or any of its children.
CompareTag	Is this game object tagged with tag ?
GetComponent	Returns the component of Type type if the game object has one attached, null if it doesn't.
GetComponentInChildren	Returns the component of Type type in the GameObject or any of its children using depth first search.
GetComponentInParent	Returns the component of Type type in the GameObject or any of its parents.
GetComponents	Returns all components of Type type in the GameObject.
GetComponentsInChildren	Returns all components of Type type in the GameObject or any of its children.
GetComponentsInParent	Returns all components of Type type in the GameObject or any of its parents.
SendMessage	Calls the method named methodName on every MonoBehaviour in this game object.
SendMessageUpwards	Calls the method named methodName on every MonoBehaviour in this game object and on every ancestor of the behaviour.
CellToLocal	Converts a cell position to local position space.
CellToLocalInterpolated	Converts an interpolated cell position in floats to local position space.
CellToWorld	Converts a cell position to world position space.
GetBoundsLocal	Returns the local bounds for a cell at the location.
GetLayoutCellCenter	Get the default center coordinate of a cell for the set layout of the Grid.

LocalToCell	Converts a local position to cell position.
LocalToCellInterpolated	Converts a local position to cell position.
LocalToWorld	Converts a local position to world position.
WorldToCell	Converts a world position to cell position.
WorldToLocal	Converts a world position to local position.
GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.

Static Methods

Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Description

Determines the orientation of tiles in the Tilemap.

Properties

XY	Orients tiles in the XY plane.
XZ	Orients tiles in the XZ plane.
YX	Orients tiles in the YX plane.
YZ	Orients tiles in the YZ plane.
ZX	Orients tiles in the ZX plane.
ZY	Orients tiles in the ZY plane.
Custom	Use a custom orientation to all tiles in the tile map.

Description

Collider for 2D physics representing shapes defined by the corresponding Tilemap.

See Also: BoxCollider2D, CircleCollider2D, EdgeCollider2D, PolygonCollider2D.

Inherited Members

Properties

enabled	Enabled Behaviours are Updated, disabled Behaviours are not.
isActiveAndEnabled	Has the Behaviour had active and enabled called?
attachedRigidbody	The Rigidbody2D attached to the Collider2D.
bounciness	Get the bounciness used by the collider.
bounds	The world space bounding area of the collider.
composite	Get the CompositeCollider2D that is available to be attached to the collider.
density	The density of the collider used to calculate its mass (when auto mass is enabled).
friction	Get the friction used by the collider.
isTrigger	Is this collider configured as a trigger?
offset	The local offset of the collider geometry.
shapeCount	The number of separate shaped regions in the collider.
sharedMaterial	The PhysicsMaterial2D that is applied to this collider.
usedByComposite	Sets whether the Collider will be used or not used by a CompositeCollider2D.
usedByEffector	Whether the collider is used by an attached effector or not.
gameObject	The game object this component is attached to. A component is always attached to a game object.
tag	The tag of this game object.
transform	The Transform attached to this GameObject.
hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.

Public Methods

Cast	Casts the collider shape into the Scene starting at the collider position ignoring the collider itself.
Distance	Calculates the minimum separation of this collider against another collider.
GetContacts	Retrieves all contact points for this collider.
IsTouching	Check whether this collider is touching the collider or not.
IsTouchingLayers	Checks whether this collider is touching any colliders on the specified layerMask or not.
OverlapCollider	Get a list of all colliders that overlap this collider.
OverlapPoint	Check if a collider overlaps a point in space.
Raycast	Casts a ray into the Scene starting at the collider position ignoring the collider itself.
BroadcastMessage	Calls the method named methodName on every MonoBehaviour in this game object or any of its children.
CompareTag	Is this game object tagged with tag ?
GetComponent	Returns the component of Type type if the game object has one attached, null if it doesn't.
GetComponentInChildren	Returns the component of Type type in the GameObject or any of its children using depth first search.
GetComponentInParent	Returns the component of Type type in the GameObject or any of its parents.
GetComponents	Returns all components of Type type in the GameObject.
GetComponentsInChildren	Returns all components of Type type in the GameObject or any of its children.
GetComponentsInParent	Returns all components of Type type in the GameObject or any of its parents.
SendMessage	Calls the method named methodName on every MonoBehaviour in this game object.
SendMessageUpwards	Calls the method named methodName on every MonoBehaviour in this game object and on every ancestor of the behaviour.
GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.

Static Methods

Destroy	Removes a gameobject, component or asset.

DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Messages

OnCollisionEnter2D	Sent when an incoming collider makes contact with this object's collider (2D physics only).
OnCollisionExit2D	Sent when a collider on another object stops touching this object's collider (2D physics only).
OnCollisionStay2D	Sent each frame where a collider on another object is touching this object's collider (2D physics only).
OnTriggerEnter2D	Sent when another object enters a trigger collider attached to this object (2D physics only).
OnTriggerExit2D	Sent when another object leaves a trigger collider attached to this object (2D physics only).
OnTriggerStay2D	Sent each frame where another object is within a trigger collider attached to this object (2D physics only).

Description

The tile map renderer is used to render the tile map marked out by a tile map component and a grid component.

This class is a script interface for a tile map renderer component.

Properties

chunkCullingBounds	Bounds used for culling of Tilemap chunks.
chunkSize	Size in number of tiles of each chunk created by the TilemapRenderer.
detectChunkCullingBounds	Returns whether the TilemapRenderer automatically detects the bounds to extend chunk culling by.
maskInteraction	Specifies how the Tilemap interacts with the masks.
maxChunkCount	Maximum number of chunks the TilemapRenderer caches in memory.
maxFrameAge	Maximum number of frames the TilemapRenderer keeps unused chunks in memory.
mode	The mode in which the TileMapRenderer batches the tiles for rendering.
sortOrder	Active sort order for the TilemapRenderer.

Inherited Members

Properties

gameObject	The game object this component is attached to. A component is always attached to a game object.
tag	The tag of this game object.
transform	The Transform attached to this GameObject.
hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.
allowOcclusionWhenDynamic	Controls if dynamic occlusion culling should be performed for this renderer.
bounds	The bounding volume of the renderer (Read Only).
enabled	Makes the rendered 3D object visible if enabled.

isPartOfStaticBatch	Has this renderer been statically batched with any other renderers?
isVisible	Is this renderer visible in any camera? (Read Only)
lightmapIndex	The index of the baked lightmap applied to this renderer.
lightmapScaleOffset	The UV scale & offset used for a lightmap.
lightProbeProxyVolumeOverride	If set, the Renderer will use the Light Probe Proxy Volume component attached to the source GameObject.
lightProbeUsage	The light probe interpolation type.
localToWorldMatrix	Matrix that transforms a point from local space into world space (Read Only).
material	Returns the first instantiated Material assigned to the renderer.
materials	Returns all the instantiated materials of this object.
motionVectorGenerationMode	Specifies the mode for motion vector rendering.
probeAnchor	If set, Renderer will use this Transform's position to find the light or reflection probe.
realtimeLightmapIndex	The index of the realtime lightmap applied to this renderer.
realtimeLightmapScaleOffset	The UV scale & offset used for a realtime lightmap.
receiveShadows	Does this object receive shadows?
reflectionProbeUsage	Should reflection probes be used for this Renderer?
rendererPriority	This value sorts renderers by priority. Lower values are rendered first and higher values are rendered last.
renderingLayerMask	Determines which rendering layer this renderer lives on.
shadowCastingMode	Does this object cast shadows?
sharedMaterial	The shared material of this object.
sharedMaterials	All the shared materials of this object.
sortingLayerID	Unique ID of the Renderer's sorting layer.
sortingLayerName	Name of the Renderer's sorting layer.
sortingOrder	Renderer's order within a sorting layer.
worldToLocalMatrix	Matrix that transforms a point from world space into local space (Read Only).

Public Methods

BroadcastMessage	Calls the method named methodName on every MonoBehaviour in this game object or any of its children.
CompareTag	Is this game object tagged with tag ?
GetComponent	Returns the component of Type type if the game object has one attached, null if it doesn't.
GetComponentInChildren	Returns the component of Type type in the GameObject or any of its children using depth first search.
GetComponentInParent	Returns the component of Type type in the GameObject or any of its parents.
GetComponents	Returns all components of Type type in the GameObject.
GetComponentsInChildren	Returns all components of Type type in the GameObject or any of its children.
GetComponentsInParent	Returns all components of Type type in the GameObject or any of its parents.
SendMessage	Calls the method named methodName on every MonoBehaviour in this game object.
SendMessageUpwards	Calls the method named methodName on every MonoBehaviour in this game object and on every ancestor of the behaviour.
GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.
GetClosestReflectionProbes	Returns an array of closest reflection probes with weights, weight shows how much influence the probe has on the renderer, this value is also used when blending between reflection probes occur.
GetMaterials	Returns all the instantiated materials of this object.
GetPropertyBlock	Get per-Renderer or per-Material property block.
GetSharedMaterials	Returns all the shared materials of this object.
HasPropertyBlock	Returns true if the Renderer has a material property block attached via SetPropertyBlock.
SetPropertyBlock	Lets you set or clear per-renderer or per-material parameter overrides.

Static Methods

Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.

DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Messages

OnBecameInvisible	OnBecameInvisible is called when the object is no longer visible by any camera.
OnBecameVisible	OnBecameVisible is called when the object became visible by any camera.

Description

Returns whether the TilemapRenderer automatically detects the bounds to extend chunk culling by.

Properties

Auto	The TilemapRenderer will automatically detect the bounds of extension by inspecting the Sprite/s used in the Tilemap.
Manual	The user adds in the values used for extend the bounds for culling of Tilemap chunks.

Description

Sort order for all tiles rendered by the TilemapRenderer.

Properties

BottomLeft	Sorts tiles for rendering starting from the tile with the lowest X and the lowest Y cell positions.
BottomRight	Sorts tiles for rendering starting from the tile with the highest X and the lowest Y cell positions.
TopLeft	Sorts tiles for rendering starting from the tile with the lowest X and the highest Y cell positions.
TopRight	Sorts tiles for rendering starting from the tile with the highest X and the lowest Y cell positions.

Description

Flags controlling behavior for the TileBase.

Properties

None	No TileFlags are set.
LockColor	TileBase locks any color set by brushes or the user.
LockTransform	TileBase locks any transform matrix set by brushes or the user.
InstantiateGameObjectRuntimeOnly	TileBase does not instantiate its associated GameObject in editor mode and instantiates it only during play mode.
LockAll	All lock flags.

Description

Playable that controls the active state of a GameObject.

Properties

gameObject	GameObject to control it active state.
postPlayback	The active state to set on the GameObject after the graph has been stopped

Inherited Members

Public Methods

OnBehaviourDelay	This function is called when the Playable play state is changed to PlayState.Delayed.
OnBehaviourPause	This function is called when the Playable play state is changed to PlayState.Paused.
OnBehaviourPlay	This function is called when the Playable play state is changed to PlayState.Playing.
OnGraphStart	This function is called when the PlayableGraph that owns this PlayableBehaviour starts.
OnGraphStop	This function is called when the PlayableGraph that owns this PlayableBehaviour stops.
OnPlayableCreate	This function is called when the Playable that owns the PlayableBehaviour is created.
OnPlayableDestroy	This function is called when the Playable that owns the PlayableBehaviour is destroyed.
PrepareData	This function is called during the PrepareData phase of the PlayableGraph.
PrepareFrame	This function is called during the PrepareFrame phase of the PlayableGraph.
ProcessFrame	This function is called during the ProcessFrame phase of the PlayableGraph.

Description

The state of a GameObject's activeness when a PlayableGraph stops.

Properties

Active	Set the GameObject to active when the PlayableGraph stops.
Inactive	Set the GameObject to inactive when the PlayableGraph stops.
Revert	Revert the GameObject to the active state it was before the PlayableGraph started.

Description

An activation track is a track that can be used to control the active state of a GameObject.

Properties

postPlaybackState	Specifies what state to leave the GameObject in after the Timeline has finished playing.

Inherited Members

Properties

hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.
duration	The playback duration in seconds of the instantiated Playable.
outputs	A description of the outputs of the instantiated Playable.
isEmpty	Does this track have any clips?
isSubTrack	Is this track a subtrack?
locked	The local locked state of the track.
lockedInHierarchy	The locked state of a track. (Read Only)
muted	Mutes the track, excluding it from the generated PlayableGraph.
outputs	A description of the outputs of the instantiated Playable.
parent	The owner of this track.
timelineAsset	The TimelineAsset this track belongs to.

Public Methods

GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.
CreatePlayable	Implement this method to have your asset inject playables into the given graph.
CreateClip	Creates a clip on the track with a custom playable asset attached, whose type is specified by T. Throws an InvalidOperationException if the specified type is not supported by the track.

CreateDefaultClip	Creates a default clip for this track, where the clip's asset type is based on any TrackClipTypeAttributes marking the track.
CreateTrackMixer	Creates a mixer used to blend playables generated by clips on the track.
GetChildTracks	The list of subtracks or child tracks attached to this track.
GetClips	Returns an enumerable list of clips owned by the track.

Protected Methods

OnAfterTrackDeserialize	Override this method to receive a callback after Unity deserializes your track.
OnBeforeTrackSerialize	Override this method to receive a callback before Unity serializes your track.
OnCreateClip	Called when a clip is created on a track. Use this method to set default values on a timeline clip, or on it's PlayableAsset.

Static Methods

Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.
CreateInstance	Creates an instance of a scriptable object.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Messages

Awake	This function is called when the ScriptableObject script is started.
OnDestroy	This function is called when the scriptable object will be destroyed.

OnDisable	This function is called when the scriptable object goes out of scope.
OnEnable	This function is called when the object is loaded.

Description

Specify what state the GameObject is set to after the Timeline has finished playing.

Properties

Active	Set the GameObject to active.
Inactive	Set the GameObject to Inactive.
Revert	Revert the GameObject to the state in was in before the Timeline was playing.
LeaveAsIs	Leave the GameObject in the state it was when the Timeline was stopped.

Implements interfaces:IPropertyPreview, ISerializationCallbackReceiver, ITimelineClipAsset

Description

A playable asset wrapper that represents a single AnimationClip clip in Timeline.

Properties

clip	The animation clip.
eulerAngles	The euler angle representation of the root motion rotation offset of the clip.
matchTargetFields	Specifies which fields should be matched when aligning offsets.
position	The root motion position offset of the clip.
removeStartOffset	Whether to make the animation clip play relative to its first keyframe.
rotation	The root motion rotation offset of the clip.
useTrackMatchFields	Specifies that matching options are defined by the track.

Public Methods

ResetOffsets	Resets the root motion offsets to default values.

Inherited Members

Properties

hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.
duration	The playback duration in seconds of the instantiated Playable.
outputs	A description of the outputs of the instantiated Playable.

Public Methods

GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.
CreatePlayable	Implement this method to have your asset inject playables into the given graph.

Static Methods

Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.
CreateInstance	Creates an instance of a scriptable object.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Messages

Awake	This function is called when the ScriptableObject script is started.
OnDestroy	This function is called when the scriptable object will be destroyed.
OnDisable	This function is called when the scriptable object goes out of scope.
OnEnable	This function is called when the object is loaded.

Description

A Timeline track used for playing back animations on an Animator.

Properties

applyAvatarMask	Specifies whether to apply the AvatarMask to the track.
avatarMask	Specifies an AvatarMask to be applied.
eulerAngles	The euler angle representation of the root motion rotation offset of the entire track.
inClipMode	Specifies whether the Animation Track has clips, or is in infinite mode.
matchTargetFields	Specifies which fields to match when aligning root motion offsets of clips.
openClipOffsetEulerAngles	The euler angle representation of the root motion rotation offset of the track when in infinite mode.
openClipOffsetPosition	This represents the root motion position offset of a track in infinite mode.
openClipOffsetRotation	This represents the root motion rotation offset of a track in infinite mode.
position	The root motion position offset of the entire track.
rotation	The root motion rotation offset of the entire track.
trackOffset	Specifies what is used to set the starting position and orientation of an Animation Track.

Public Methods

CreateClip	Creates a TimelineClip using an AnimationClip.
CreateRecordableClip	Creates a TimelineClip, AnimationPlayableAsset and an AnimationClip. Use this clip to record in a timeline.

Inherited Members

Properties

hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.

duration	The playback duration in seconds of the instantiated Playable.
outputs	A description of the outputs of the instantiated Playable.
isEmpty	Does this track have any clips?
isSubTrack	Is this track a subtrack?
locked	The local locked state of the track.
lockedInHierarchy	The locked state of a track. (Read Only)
muted	Mutes the track, excluding it from the generated PlayableGraph.
outputs	A description of the outputs of the instantiated Playable.
parent	The owner of this track.
timelineAsset	The TimelineAsset this track belongs to.

Public Methods

GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.
CreatePlayable	Implement this method to have your asset inject playables into the given graph.
CreateClip	Creates a clip on the track with a custom playable asset attached, whose type is specified by T. Throws an InvalidOperationException if the specified type is not supported by the track.
CreateDefaultClip	Creates a default clip for this track, where the clip's asset type is based on any TrackClipTypeAttributes marking the track.
CreateTrackMixer	Creates a mixer used to blend playables generated by clips on the track.
GetChildTracks	The list of subtracks or child tracks attached to this track.
GetClips	Returns an enumerable list of clips owned by the track.

Protected Methods

OnAfterTrackDeserialize	Override this method to receive a callback after Unity deserializes your track.
OnBeforeTrackSerialize	Override this method to receive a callback before Unity serializes your track.
OnCreateClip	Called when a clip is created on a track. Use this method to set default values on a timeline clip, or on it's PlayableAsset.

Static Methods

Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.
CreateInstance	Creates an instance of a scriptable object.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Messages

Awake	This function is called when the ScriptableObject script is started.
OnDestroy	This function is called when the scriptable object will be destroyed.
OnDisable	This function is called when the scriptable object goes out of scope.
OnEnable	This function is called when the object is loaded.

Implements interfaces:ITimelineClipAsset

Description

PlayableAsset wrapper for an AudioClip in Timeline.

Properties

clip	The audio clip this asset will generate a player for.
loop	Whether the audio clip loops.

Inherited Members

Properties

hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.
duration	The playback duration in seconds of the instantiated Playable.
outputs	A description of the outputs of the instantiated Playable.

Public Methods

GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.
CreatePlayable	Implement this method to have your asset inject playables into the given graph.

Static Methods

Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.

CreateInstance	Creates an instance of a scriptable object.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Messages

Awake	This function is called when the ScriptableObject script is started.
OnDestroy	This function is called when the scriptable object will be destroyed.
OnDisable	This function is called when the scriptable object goes out of scope.
OnEnable	This function is called when the object is loaded.

Description

A Timeline track that can play AudioClips.

Public Methods

CreateClip	Create an TimelineClip for playing an AudioClip on this track.

Inherited Members

Properties

hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.
duration	The playback duration in seconds of the instantiated Playable.
outputs	A description of the outputs of the instantiated Playable.
isEmpty	Does this track have any clips?
isSubTrack	Is this track a subtrack?
locked	The local locked state of the track.
lockedInHierarchy	The locked state of a track. (Read Only)
muted	Mutes the track, excluding it from the generated PlayableGraph.
outputs	A description of the outputs of the instantiated Playable.
parent	The owner of this track.
timelineAsset	The TimelineAsset this track belongs to.

Public Methods

GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.
CreatePlayable	Implement this method to have your asset inject playables into the given graph.
CreateClip	Creates a clip on the track with a custom playable asset attached, whose type is specified by T. Throws an InvalidOperationException if the specified type is not supported by the track.

CreateDefaultClip	Creates a default clip for this track, where the clip's asset type is based on any TrackClipTypeAttributes marking the track.
CreateTrackMixer	Creates a mixer used to blend playables generated by clips on the track.
GetChildTracks	The list of subtracks or child tracks attached to this track.
GetClips	Returns an enumerable list of clips owned by the track.

Protected Methods

OnAfterTrackDeserialize	Override this method to receive a callback after Unity deserializes your track.
OnBeforeTrackSerialize	Override this method to receive a callback before Unity serializes your track.
OnCreateClip	Called when a clip is created on a track. Use this method to set default values on a timeline clip, or on it's PlayableAsset.

Static Methods

Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.
CreateInstance	Creates an instance of a scriptable object.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Messages

Awake	This function is called when the ScriptableObject script is started.
OnDestroy	This function is called when the scriptable object will be destroyed.

OnDisable	This function is called when the scriptable object goes out of scope.
OnEnable	This function is called when the object is loaded.

Implements interfaces:IPropertyPreview, ITimelineClipAsset

Description

Asset that generates playables for controlling time-related elements on a GameObject.

This playable optionally controls ParticleSystem, PlayableDirectors and Activation from a GameObject. Users can also specify a Prefab that will be instantiated, and parented to the the sourceObject if specified.

Properties

active	Indicate if the playable will use Activation.
particleRandomSeed	Let the particle systems behave the same way on each execution.
postPlayback	Indicates the active state of the gameObject when the Timeline is stopped.
prefabGameObject	Prefab object that will be instantiated.
searchHierarchy	Indicate whether to search the entire hierachy for controlable components.
sourceGameObject	GameObject in the Scene to control, or the parent of the instantiated Prefab.
updateDirector	Indicate if user wants to control PlayableDirectors.
updateITimeControl	Indicates that whether Monobehaviours implementing ITimeControl on the gameObject will be controlled.
updateParticle	Indicates if user wants to control ParticleSystems.

Inherited Members

Properties

hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.
duration	The playback duration in seconds of the instantiated Playable.
outputs	A description of the outputs of the instantiated Playable.

Public Methods

GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.

CreatePlayable	Implement this method to have your asset inject playables into the given graph.

Static Methods

Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.
CreateInstance	Creates an instance of a scriptable object.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Messages

Awake	This function is called when the ScriptableObject script is started.
OnDestroy	This function is called when the scriptable object will be destroyed.
OnDisable	This function is called when the scriptable object goes out of scope.
OnEnable	This function is called when the object is loaded.

Description

A Track whose clips control time-related elements on a GameObject.

Inherited Members

Properties

hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.
duration	The playback duration in seconds of the instantiated Playable.
outputs	A description of the outputs of the instantiated Playable.
isEmpty	Does this track have any clips?
isSubTrack	Is this track a subtrack?
locked	The local locked state of the track.
lockedInHierarchy	The locked state of a track. (Read Only)
muted	Mutes the track, excluding it from the generated PlayableGraph.
outputs	A description of the outputs of the instantiated Playable.
parent	The owner of this track.
timelineAsset	The TimelineAsset this track belongs to.

Public Methods

GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.
CreatePlayable	Implement this method to have your asset inject playables into the given graph.
CreateClip	Creates a clip on the track with a custom playable asset attached, whose type is specified by T. Throws an InvalidOperationException if the specified type is not supported by the track.
CreateDefaultClip	Creates a default clip for this track, where the clip's asset type is based on any TrackClipTypeAttributes marking the track.
CreateTrackMixer	Creates a mixer used to blend playables generated by clips on the track.

GetChildTracks	The list of subtracks or child tracks attached to this track.
GetClips	Returns an enumerable list of clips owned by the track.

Protected Methods

OnAfterTrackDeserialize	Override this method to receive a callback after Unity deserializes your track.
OnBeforeTrackSerialize	Override this method to receive a callback before Unity serializes your track.
OnCreateClip	Called when a clip is created on a track. Use this method to set default values on a timeline clip, or on it's PlayableAsset.

Static Methods

Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.
CreateInstance	Creates an instance of a scriptable object.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Messages

Awake	This function is called when the ScriptableObject script is started.
OnDestroy	This function is called when the scriptable object will be destroyed.
OnDisable	This function is called when the scriptable object goes out of scope.
OnEnable	This function is called when the object is loaded.

Description

Playable used to control other PlayableDirectors.

This playable is used to control other PlayableDirector components from a Timeline sequence.

Properties

director	The PlayableDirector to control.

Inherited Members

Public Methods

OnBehaviourDelay	This function is called when the Playable play state is changed to PlayState.Delayed.
OnBehaviourPause	This function is called when the Playable play state is changed to PlayState.Paused.
OnBehaviourPlay	This function is called when the Playable play state is changed to PlayState.Playing.
OnGraphStart	This function is called when the PlayableGraph that owns this PlayableBehaviour starts.
OnGraphStop	This function is called when the PlayableGraph that owns this PlayableBehaviour stops.
OnPlayableCreate	This function is called when the Playable that owns the PlayableBehaviour is created.
OnPlayableDestroy	This function is called when the Playable that owns the PlayableBehaviour is destroyed.
PrepareData	This function is called during the PrepareData phase of the PlayableGraph.
PrepareFrame	This function is called during the PrepareFrame phase of the PlayableGraph.
ProcessFrame	This function is called during the ProcessFrame phase of the PlayableGraph.

Description

A group track is a container that allows tracks to be arranged in a hierarchical manner.

Inherited Members

Properties

hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.
duration	The playback duration in seconds of the instantiated Playable.
outputs	A description of the outputs of the instantiated Playable.
isEmpty	Does this track have any clips?
isSubTrack	Is this track a subtrack?
locked	The local locked state of the track.
lockedInHierarchy	The locked state of a track. (Read Only)
muted	Mutes the track, excluding it from the generated PlayableGraph.
outputs	A description of the outputs of the instantiated Playable.
parent	The owner of this track.
timelineAsset	The TimelineAsset this track belongs to.

Public Methods

GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.
CreatePlayable	Implement this method to have your asset inject playables into the given graph.
CreateClip	Creates a clip on the track with a custom playable asset attached, whose type is specified by T. Throws an InvalidOperationException if the specified type is not supported by the track.
CreateDefaultClip	Creates a default clip for this track, where the clip's asset type is based on any TrackClipTypeAttributes marking the track.
CreateTrackMixer	Creates a mixer used to blend playables generated by clips on the track.

GetChildTracks	The list of subtracks or child tracks attached to this track.
GetClips	Returns an enumerable list of clips owned by the track.

Protected Methods

OnAfterTrackDeserialize	Override this method to receive a callback after Unity deserializes your track.
OnBeforeTrackSerialize	Override this method to receive a callback before Unity serializes your track.
OnCreateClip	Called when a clip is created on a track. Use this method to set default values on a timeline clip, or on it's PlayableAsset.

Static Methods

Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.
CreateInstance	Creates an instance of a scriptable object.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Messages

Awake	This function is called when the ScriptableObject script is started.
OnDestroy	This function is called when the scriptable object will be destroyed.
OnDisable	This function is called when the scriptable object goes out of scope.
OnEnable	This function is called when the object is loaded.

Description

Playable that synchronizes a particle system simulation.

Properties

particleSystem	Which particle system to control.

Inherited Members

Public Methods

OnBehaviourDelay	This function is called when the Playable play state is changed to PlayState.Delayed.
OnBehaviourPause	This function is called when the Playable play state is changed to PlayState.Paused.
OnBehaviourPlay	This function is called when the Playable play state is changed to PlayState.Playing.
OnGraphStart	This function is called when the PlayableGraph that owns this PlayableBehaviour starts.
OnGraphStop	This function is called when the PlayableGraph that owns this PlayableBehaviour stops.
OnPlayableCreate	This function is called when the Playable that owns the PlayableBehaviour is created.
OnPlayableDestroy	This function is called when the Playable that owns the PlayableBehaviour is destroyed.
PrepareData	This function is called during the PrepareData phase of the PlayableGraph.
PrepareFrame	This function is called during the PrepareFrame phase of the PlayableGraph.
ProcessFrame	This function is called during the ProcessFrame phase of the PlayableGraph.

Description

A PlayableTrack is a track whose clips are custom playables.

This is a track that can contain PlayableAssets that are found in the project and do not have their own specified track type.

Inherited Members

Properties

hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.
duration	The playback duration in seconds of the instantiated Playable.
outputs	A description of the outputs of the instantiated Playable.
isEmpty	Does this track have any clips?
isSubTrack	Is this track a subtrack?
locked	The local locked state of the track.
lockedInHierarchy	The locked state of a track. (Read Only)
muted	Mutes the track, excluding it from the generated PlayableGraph.
outputs	A description of the outputs of the instantiated Playable.
parent	The owner of this track.
timelineAsset	The TimelineAsset this track belongs to.

Public Methods

GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.
CreatePlayable	Implement this method to have your asset inject playables into the given graph.
CreateClip	Creates a clip on the track with a custom playable asset attached, whose type is specified by T. Throws an InvalidOperationException if the specified type is not supported by the track.
CreateDefaultClip	Creates a default clip for this track, where the clip's asset type is based on any TrackClipTypeAttributes marking the track.

CreateTrackMixer	Creates a mixer used to blend playables generated by clips on the track.
GetChildTracks	The list of subtracks or child tracks attached to this track.
GetClips	Returns an enumerable list of clips owned by the track.

Protected Methods

OnAfterTrackDeserialize	Override this method to receive a callback after Unity deserializes your track.
OnBeforeTrackSerialize	Override this method to receive a callback before Unity serializes your track.
OnCreateClip	Called when a clip is created on a track. Use this method to set default values on a timeline clip, or on it's PlayableAsset.

Static Methods

Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.
CreateInstance	Creates an instance of a scriptable object.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Messages

Awake	This function is called when the ScriptableObject script is started.
OnDestroy	This function is called when the scriptable object will be destroyed.
OnDisable	This function is called when the scriptable object goes out of scope.

OnEnable	This function is called when the object is loaded.

Description

Playable that controls and instantiates a Prefab.

Properties

prefabInstance	The instance of the Prefab that has been generated for this playable.

Inherited Members

Public Methods

OnBehaviourDelay	This function is called when the Playable play state is changed to PlayState.Delayed.
OnBehaviourPause	This function is called when the Playable play state is changed to PlayState.Paused.
OnBehaviourPlay	This function is called when the Playable play state is changed to PlayState.Playing.
OnGraphStart	This function is called when the PlayableGraph that owns this PlayableBehaviour starts.
OnGraphStop	This function is called when the PlayableGraph that owns this PlayableBehaviour stops.
OnPlayableCreate	This function is called when the Playable that owns the PlayableBehaviour is created.
OnPlayableDestroy	This function is called when the Playable that owns the PlayableBehaviour is destroyed.
PrepareData	This function is called during the PrepareData phase of the PlayableGraph.
PrepareFrame	This function is called during the PrepareFrame phase of the PlayableGraph.
ProcessFrame	This function is called during the ProcessFrame phase of the PlayableGraph.

Implements interfaces:IPropertyPreview, ISerializationCallbackReceiver, ITimelineClipAsset

Description

A PlayableAsset that represents a timeline.

Properties

duration	The length in seconds of the timeline.
durationMode	How the duration of a timeline is calculated.
fixedDuration	The length of the timeline when the durationMode is set to fixed length.
outputTrackCount	Returns the the number of output tracks in the Timeline.
rootTrackCount	Returns the number of tracks at the root level of the timeline.

Public Methods

CreateTrack	Allows you to create a track and add it to the Timeline.
DeleteClip	Delete a clip from this timeline.
DeleteTrack	Deletes a track from a timeline, including all clips and subtracks.
GetOutputTrack	Retrives the output track from the given index.
GetOutputTracks	Gets a list of all output tracks in the Timeline.
GetRootTrack	Retrieves at root track at the specified index.
GetRootTracks	Get an enumerable list of all root tracks.

Inherited Members

Properties

hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.
duration	The playback duration in seconds of the instantiated Playable.
outputs	A description of the outputs of the instantiated Playable.

Public Methods

GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.
CreatePlayable	Implement this method to have your asset inject playables into the given graph.

Static Methods

Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.
CreateInstance	Creates an instance of a scriptable object.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Messages

Awake	This function is called when the ScriptableObject script is started.
OnDestroy	This function is called when the scriptable object will be destroyed.
OnDisable	This function is called when the scriptable object goes out of scope.
OnEnable	This function is called when the object is loaded.

Description

How the duration of the timeline is determined.

Properties

BasedOnClips	The duration of the timeline is determined based on the clips present.
FixedLength	The duration of the timeline is a fixed length.

Implements interfaces:ISerializationCallbackReceiver

Description

Represents a clip on the timeline.

Properties

animationClip	An AnimationClip containing animated parameters.
asset	Reference to a serializable IPlayableAsset representing the specialization of the clip.
clipCaps	Feature capabilities supported by this clip.
clipIn	Time to start playing the clip at.
duration	The length in seconds of the clip.
end	The end time of the clip.
extrapolatedDuration	The length of the clip including extrapolation.
extrapolatedStart	The start time of the clip when extrapolation is considered.
hasBlendIn	Does the clip have blend in?
hasBlendOut	Does the clip have a non-zero blend out?
hasPostExtrapolation	Is the clip being extrapolated past it's end time?
hasPreExtrapolation	Is the clip being extrapolated before it's start time?
parentTrack	Returns the parent track where this clip is attached.
postExtrapolationMode	The extrapolation mode for time beyond the end of the clip.
preExtrapolationMode	The extrapolation mode for the time before the start of the clip.
start	The start time of the clip.
timeScale	The time scale of the clip.

Public Methods

ToLocalTime	Converts from global time to a clips local time.
ToLocalTimeUnbound	Converts from global time to a clips local time, ignoring extrapolation.

Description

How the clip handles time outside it's start and end range.

Properties

None	No extrapolation is applied.
Hold	Hold the time at the end value of the clip.
Loop	Loop time values outside the start/end range.
Continue	Lets the underlying asset handle extrapolated time values.

Description

Playable generated by a Timeline.

Inherited Members

Public Methods

OnBehaviourDelay	This function is called when the Playable play state is changed to PlayState.Delayed.
OnBehaviourPause	This function is called when the Playable play state is changed to PlayState.Paused.
OnBehaviourPlay	This function is called when the Playable play state is changed to PlayState.Playing.
OnGraphStart	This function is called when the PlayableGraph that owns this PlayableBehaviour starts.
OnGraphStop	This function is called when the PlayableGraph that owns this PlayableBehaviour stops.
OnPlayableCreate	This function is called when the Playable that owns the PlayableBehaviour is created.
OnPlayableDestroy	This function is called when the Playable that owns the PlayableBehaviour is destroyed.
PrepareData	This function is called during the PrepareData phase of the PlayableGraph.
PrepareFrame	This function is called during the PrepareFrame phase of the PlayableGraph.
ProcessFrame	This function is called during the ProcessFrame phase of the PlayableGraph.

Implements interfaces:IPropertyPreview, ISerializationCallbackReceiver

Description

A PlayableAsset Representing a track inside a timeline.

Properties

isEmpty	Does this track have any clips?
isSubTrack	Is this track a subtrack?
locked	The local locked state of the track.
lockedInHierarchy	The locked state of a track. (Read Only)
muted	Mutes the track, excluding it from the generated PlayableGraph.
outputs	A description of the outputs of the instantiated Playable.
parent	The owner of this track.
timelineAsset	The TimelineAsset this track belongs to.

Public Methods

CreateClip	Creates a clip on the track with a custom playable asset attached, whose type is specified by T. Throws an InvalidOperationException if the specified type is not supported by the track.
CreateDefaultClip	Creates a default clip for this track, where the clip's asset type is based on any TrackClipTypeAttributes marking the track.
CreateTrackMixer	Creates a mixer used to blend playables generated by clips on the track.
GetChildTracks	The list of subtracks or child tracks attached to this track.
GetClips	Returns an enumerable list of clips owned by the track.

Protected Methods

OnAfterTrackDeserialize	Override this method to receive a callback after Unity deserializes your track.
OnBeforeTrackSerialize	Override this method to receive a callback before Unity serializes your track.
OnCreateClip	Called when a clip is created on a track. Use this method to set default values on a timeline clip, or on it's PlayableAsset.

Inherited Members

Properties

hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.
duration	The playback duration in seconds of the instantiated Playable.
outputs	A description of the outputs of the instantiated Playable.

Public Methods

GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.
CreatePlayable	Implement this method to have your asset inject playables into the given graph.

Static Methods

Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.
CreateInstance	Creates an instance of a scriptable object.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Messages

Awake	This function is called when the ScriptableObject script is started.

OnDestroy	This function is called when the scriptable object will be destroyed.
OnDisable	This function is called when the scriptable object goes out of scope.
OnEnable	This function is called when the object is loaded.

Description

Extension methods for Track Assets.

Static Methods

GetGroup	Gets the GroupTrack this track belongs to.
SetGroup	Assigns the track to the specified group track.

Description

Interface used to inform the Timeline Editor about potential property modifications that may occur while previewing.

Public Methods

AddFromClip	Add property modifications modified by an animation clip.
AddFromClips	Add property modifications specified by a list of animation clips.
AddFromName	Add property modifications using the serialized property name.
AddObjectProperties	Add property modifications for a ScriptableObject.
PopActiveGameObject	Removes the active GameObject from the modification stack, restoring the previous value.
PushActiveGameObject	Sets the active game object for subsequent property modifications.

Description

Implement this interface in a PlayableAsset to specify which properties will be modified when Timeline is in preview mode.

Public Methods

GatherProperties	Called by the Timeline Editor to gather properties requiring preview.

Description

Interface that can be implemented by MonoBehaviours indicating that they receive time-related control calls from a PlayableGraph.

Implementing this interface on MonoBehaviours attached to GameObjects under control by control-tracks will cause them to be notified when associated Timeline clips are active.

Public Methods

OnControlTimeStart	Called when the associated Timeline clip becomes active.
OnControlTimeStop	Called when the associated Timeline clip becomes deactivated.
SetTime	Called each frame the Timeline clip is active.

Description

Implement this interface to support advanced features of timeline clips.

Properties

clipCaps	Returns a description of the features supported by clips representing playables implementing this interface.

Description

Describes the timeline features supported by clips representing this playable.

Properties

None	No features are supported.
Looping	The clip representing this playable supports loops.
Extrapolation	The clip representing this playable supports clip extrapolation.
ClipIn	The clip representing the playable supports initial local times greater than zero.
SpeedMultiplier	The clip representing this playable supports time scaling.
Blending	The clip representing the playable supports blending between clips.
All	All features are supported.

Description

Specify these options when using a binding type for a custom TrackAsset.

Properties

None	Specifies no options.
AllowCreateComponent	Specifies the automatic creation of component types, when required.
All	Specifies all options. This is the default.

Description

Attribute to apply to a ScriptPlayable class or property to indicate that it is not animatable.

Description

Specifies the type of object that should be bound to a TrackAsset.

Use this attribute when creating Custom Tracks to specify the type of object the track requires a binding to.

```
using UnityEngine;
using UnityEngine.Timeline;


[TrackBindingType(typeof(Light), TrackBindingFlags.AllowCreateComponent)]
public class LightTrack : TrackAsset
{
}
```

Properties

flags	Options related to the track binding.
type	The type of object that the track requires a binding to.

Description

Specifies the type of PlayableAsset that this track can create clips representing.

This is required for custom track assets to specify the type of clips to create.

Constructors

TrackClipTypeAttribute	Specifies the type of PlayableAsset that this track can create clips representing.

Description

Attribute used to specify the color of the track and its clips inside the Timeline Editor.

Constructors

TrackColorAttribute	Specify the track color using [0-1] R,G,B values.

Description

Interface into tvOS specific functionality.

Static Properties

advertisingIdentifier	Advertising ID.
advertisingTrackingEnabled	Is advertising tracking enabled.
generation	The generation of the device. (Read Only)
systemVersion	iOS version.
vendorIdentifier	Vendor ID.

Static Methods

ResetNoBackupFlag	Reset "no backup" file flag: file will be synced with iCloud/iTunes backup and can be deleted by OS in low storage situations.
SetNoBackupFlag	Set file flag to be excluded from iCloud/iTunes backup.

Description

A class for Apple TV remote input configuration.

Static Properties

allowExitToHome	Configures how "Menu" button behaves on Apple TV Remote. If this property is set to true hitting "Menu" on Remote will exit to system home screen. When this property is false current application is responsible for handling "Menu" button. It is recommended to set this property to true on top level menus of your application.
allowRemoteRotation	Configures if Apple TV Remote should autorotate all the inputs when Remote is being held in horizontal orientation. Default is false.
reportAbsoluteDpadValues	Configures how touches are mapped to analog joystick axes in relative or absolute values. If set to true it will return +1 on Horizontal axis when very far right is being touched on Remote touch aread (and -1 when very left area is touched correspondingly). The same applies for Vertical axis too. When this property is set to false player should swipe instead of touching specific area of remote to generate Horizontal or Vertical input.
touchesEnabled	Disables Apple TV Remote touch propagation to Unity Input.touches API. Useful for 3rd party frameworks, which do not respect Touch.type == Indirect. Default is false.

Description

iOS device generation.

Properties

AppleTV1Gen	First generation Apple TV.
AppleTV2Gen	Second generation (4K) Apple TV.

Description

Sprite Atlas is an asset created within Unity. It is part of the built-in sprite packing solution.

A Sprite Atlas stores a list of packable assets. A packable asset is either a Sprite, Texture2D of TextureImporterType.Sprite or Folder. Before the packing process begins, these packable assets will be grouped and traversed to gather all the sprites from them. These will be used in the packing process. At runtime, these sprites can be enumerated via the Sprite Atlas (See Also: SpriteAtlas.GetSprites).

It also provides dedicated texture settings in the inspector for the packed texture. The original texture settings of the sprite will have no effect on the packed texture.

By default, Sprite Atlas will be referenced by the sprite and be available at runtime. This means that the sprite will be able to acquire the packed texture from the Sprite Atlas when loaded. A Sprite can be loaded without referencing any Sprite Atlas. A Sprite loaded this way will render as invisible since there is no texture. A reference to a Sprite Atlas can be added later. See Also: SpriteAtlasManager.

Sprite Atlas variants can be created by assigning another Sprite Atlas object as the master. Variants will not repack a new texture from the packable list. Instead of this, variants will duplicate the master's packed texture and downscale it according to a user-defined ratio and save this scaled texture.

Properties

isVariant	Return true if this SpriteAtlas is a variant.
spriteCount	Get the total number of Sprite packed into this atlas.
tag	Get the tag of this SpriteAtlas.

Public Methods

CanBindTo	Return true if Sprite is packed into this SpriteAtlas.
GetSprite	Clone the first Sprite in this atlas that matches the name packed in this atlas and return it.
GetSprites	Clone all the Sprite in this atlas and fill them into the supplied array.

Inherited Members

Properties

hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.

Public Methods

GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.

Static Methods

Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Description

Manages SpriteAtlas during runtime.

A Sprite can be loaded without referencing any Sprite Atlas, thus having no texture. It will be invisible until the user registers an atlas to the Sprite by listening to the SpriteAtlasManager.atlasRequested callback. When triggered, it will pass in the atlas tag and a System.Action which will take in an atlas object.

Events

atlasRegistered	Trigger when a SpriteAtlas is registered via invoking the callback in SpriteAtlasManager.atlasRequested.
atlasRequested	Trigger when any Sprite was bound to SpriteAtlas but couldn't locate the atlas asset during runtime.

Description

Structure to store the state of an animation transition on a Selectable.

Properties

disabledTrigger	Trigger to send to animator when entering disabled state.
highlightedTrigger	Trigger to send to animator when entering highlighted state.
normalTrigger	Trigger to send to animator when entering normal state.
pressedTrigger	Trigger to send to animator when entering pressed state.

Implements interfaces:ILayoutController, ILayoutSelfController

Description

Resizes a RectTransform to fit a specified aspect ratio.

Properties

aspectMode	The mode to use to enforce the aspect ratio.
aspectRatio	The aspect ratio to enforce. This means width divided by height.

Public Methods

SetLayoutHorizontal	Method called by the layout system.
SetLayoutVertical	Method called by the layout system.

Protected Methods

OnDisable	See MonoBehaviour.OnDisable.
SetDirty	Mark the AspectRatioFitter as dirty.

Inherited Members

Properties

enabled	Enabled Behaviours are Updated, disabled Behaviours are not.
isActiveAndEnabled	Has the Behaviour had active and enabled called?
gameObject	The game object this component is attached to. A component is always attached to a game object.
tag	The tag of this game object.
transform	The Transform attached to this GameObject.
runInEditMode	Allow a specific instance of a MonoBehaviour to run in edit mode (only available in the editor).
useGUILayout	Disabling this lets you skip the GUI layout phase.
hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?

name	The name of the object.

Public Methods

BroadcastMessage	Calls the method named methodName on every MonoBehaviour in this game object or any of its children.
CompareTag	Is this game object tagged with tag ?
GetComponent	Returns the component of Type type if the game object has one attached, null if it doesn't.
GetComponentInChildren	Returns the component of Type type in the GameObject or any of its children using depth first search.
GetComponentInParent	Returns the component of Type type in the GameObject or any of its parents.
GetComponents	Returns all components of Type type in the GameObject.
GetComponentsInChildren	Returns all components of Type type in the GameObject or any of its children.
GetComponentsInParent	Returns all components of Type type in the GameObject or any of its parents.
SendMessage	Calls the method named methodName on every MonoBehaviour in this game object.
SendMessageUpwards	Calls the method named methodName on every MonoBehaviour in this game object and on every ancestor of the behaviour.
CancelInvoke	Cancels all Invoke calls on this MonoBehaviour.
Invoke	Invokes the method methodName in time seconds.
InvokeRepeating	Invokes the method methodName in time seconds, then repeatedly every repeatRate seconds.
IsInvoking	Is any invoke on methodName pending?
StartCoroutine	Starts a coroutine.
StopAllCoroutines	Stops all coroutines running on this behaviour.
StopCoroutine	Stops the first coroutine named methodName, or the coroutine stored in routine running on this behaviour.
GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.
IsActive	Returns true if the GameObject and the Component are active.
IsDestroyed	Returns true if the native representation of the behaviour has been destroyed.

Protected Methods

Awake	See MonoBehaviour.Awake.
OnBeforeTransformParentChanged	See MonoBehaviour.OnBeforeTransformParentChanged.
OnCanvasGroupChanged	See MonoBehaviour.OnCanvasGroupChanged.
OnCanvasHierarchyChanged	Called when the state of the parent Canvas is changed.
OnDestroy	See MonoBehaviour.OnDestroy.
OnDidApplyAnimationProperties	See LayoutGroup.OnDidApplyAnimationProperties.
OnEnable	See MonoBehaviour.OnEnable.
OnRectTransformDimensionsChange	This callback is called if an associated RectTransform has its dimensions changed.
OnTransformParentChanged	See MonoBehaviour.OnRectTransformParentChanged.
OnValidate	See MonoBehaviour.OnValidate.
Reset	See MonoBehaviour.Reset.
Start	See MonoBehaviour.Start.

Static Methods

print	Logs message to the Unity Console (identical to Debug.Log).
Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Messages

Awake	Awake is called when the script instance is being loaded.
FixedUpdate	This function is called every fixed framerate frame, if the MonoBehaviour is enabled.
LateUpdate	LateUpdate is called every frame, if the Behaviour is enabled.
OnAnimatorIK	Callback for setting up animation IK (inverse kinematics).
OnAnimatorMove	Callback for processing animation movements for modifying root motion.
OnApplicationFocus	Sent to all GameObjects when the player gets or loses focus.
OnApplicationPause	Sent to all GameObjects when the application pauses.
OnApplicationQuit	Sent to all game objects before the application quits.
OnAudioFilterRead	If OnAudioFilterRead is implemented, Unity will insert a custom filter into the audio DSP chain.
OnBecameInvisible	OnBecameInvisible is called when the renderer is no longer visible by any camera.
OnBecameVisible	OnBecameVisible is called when the renderer became visible by any camera.
OnCollisionEnter	OnCollisionEnter is called when this collider/rigidbody has begun touching another rigidbody/collider.
OnCollisionEnter2D	Sent when an incoming collider makes contact with this object's collider (2D physics only).
OnCollisionExit	OnCollisionExit is called when this collider/rigidbody has stopped touching another rigidbody/collider.
OnCollisionExit2D	Sent when a collider on another object stops touching this object's collider (2D physics only).
OnCollisionStay	:ref::OnCollisionStay is called once per frame for every collider/rigidbody that is touching rigidbody/collider.
OnCollisionStay2D	Sent each frame where a collider on another object is touching this object's collider (2D physics only).
OnConnectedToServer	Called on the client when you have successfully connected to a server.
OnControllerColliderHit	OnControllerColliderHit is called when the controller hits a collider while performing a Move.
OnDestroy	Destroying the attached Behaviour will result in the game or Scene

	receiving OnDestroy.
OnDisconnectedFromServer	Called on the client when the connection was lost or you disconnected from the server.
OnDrawGizmos	Implement OnDrawGizmos if you want to draw gizmos that are also pickable and always drawn.
OnDrawGizmosSelected	Implement OnDrawGizmosSelected to draw a gizmo if the object is selected.
OnEnable	This function is called when the object becomes enabled and active.
OnFailedToConnect	Called on the client when a connection attempt fails for some reason.
OnFailedToConnectToMasterServer	Called on clients or servers when there is a problem connecting to the MasterServer.
OnGUI	OnGUI is called for rendering and handling GUI events.
OnJointBreak	Called when a joint attached to the same game object broke.
OnJointBreak2D	Called when a Joint2D attached to the same game object breaks.
OnMasterServerEvent	Called on clients or servers when reporting events from the MasterServer.
OnMouseDown	OnMouseDown is called when the user has pressed the mouse button while over the GUIElement or Collider.
OnMouseDrag	OnMouseDrag is called when the user has clicked on a GUIElement or Collider and is still holding down the mouse.
OnMouseEnter	Called when the mouse enters the GUIElement or Collider.
OnMouseExit	Called when the mouse is not any longer over the GUIElement or Collider.
OnMouseOver	Called every frame while the mouse is over the GUIElement or Collider.
OnMouseUp	OnMouseUp is called when the user has released the mouse button.
OnMouseUpAsButton	OnMouseUpAsButton is only called when the mouse is released over the same GUIElement or Collider as it was pressed.
OnNetworkInstantiate	Called on objects which have been network instantiated with Network.Instantiate.
OnParticleCollision	OnParticleCollision is called when a particle hits a Collider.
OnParticleSystemStopped	OnParticleSystemStopped is called when all particles in the system have died, and no new particles will be born. New particles cease to be

	created either after Stop is called, or when the duration property of a non-looping system has been exceeded.
OnParticleTrigger	OnParticleTrigger is called when any particles in a particle system meet the conditions in the trigger module.
OnPlayerConnected	Called on the server whenever a new player has successfully connected.
OnPlayerDisconnected	Called on the server whenever a player disconnected from the server.
OnPostRender	OnPostRender is called after a camera finished rendering the Scene.
OnPreCull	OnPreCull is called before a camera culls the Scene.
OnPreRender	OnPreRender is called before a camera starts rendering the Scene.
OnRenderImage	OnRenderImage is called after all rendering is complete to render image.
OnRenderObject	OnRenderObject is called after camera has rendered the Scene.
OnSerializeNetworkView	Used to customize synchronization of variables in a script watched by a network view.
OnServerInitialized	Called on the server whenever a Network.InitializeServer was invoked and has completed.
OnTransformChildrenChanged	This function is called when the list of children of the transform of the GameObject has changed.
OnTransformParentChanged	This function is called when the parent property of the transform of the GameObject has changed.
OnTriggerEnter	OnTriggerEnter is called when the Collider other enters the trigger.
OnTriggerEnter2D	Sent when another object enters a trigger collider attached to this object (2D physics only).
OnTriggerExit	OnTriggerExit is called when the Collider other has stopped touching the trigger.
OnTriggerExit2D	Sent when another object leaves a trigger collider attached to this object (2D physics only).
OnTriggerStay	OnTriggerStay is called once per physics update for every Collider other that is touching the trigger.
OnTriggerStay2D	Sent each frame where another object is within a trigger collider attached to this object (2D physics only).
OnValidate	This function is called when the script is loaded or a value is changed in the inspector (Called in the editor only).

OnWillRenderObject	OnWillRenderObject is called for each camera if the object is visible and not a UI element.
Reset	Reset to default values.
Start	Start is called on the frame when a script is enabled just before any of the Update methods are called the first time.
Update	Update is called every frame, if the MonoBehaviour is enabled.

Description

Specifies a mode to use to enforce an aspect ratio.

Properties

None	The aspect ratio is not enforced.
WidthControlsHeight	Changes the height of the rectangle to match the aspect ratio.
HeightControlsWidth	Changes the width of the rectangle to match the aspect ratio.
FitInParent	Sizes the rectangle such that it's fully contained within the parent rectangle.
EnvelopeParent	Sizes the rectangle such that the parent rectangle is fully contained within.

Implements interfaces:IMeshModifier

Description

Base class for effects that modify the generated Mesh.

```
using UnityEngine;
using UnityEngine.UI;


public class PositionAsUV1 : BaseMeshEffect
{
    protected PositionAsUV1()
    {}


    public override void ModifyMesh(Mesh mesh)
    {
        if (!IsActive())
            return;


        var verts = mesh.vertices.ToList();
        var uvs = ListPool<Vector2>.Get();


        for (int i = 0; i < verts.Count; i++)
        {
            var vert = verts[i];
            uvs.Add(new Vector2(verts[i].x, verts[i].y));
            verts[i] = vert;
        }
        mesh.SetUVs(1, uvs);
        ListPool<Vector2>.Release(uvs);
    }
}
```

Public Methods

ModifyMesh	See:IMeshModifier.

Protected Methods

OnDisable	See MonoBehaviour.OnDisable.

Inherited Members

Properties

enabled	Enabled Behaviours are Updated, disabled Behaviours are not.
isActiveAndEnabled	Has the Behaviour had active and enabled called?
gameObject	The game object this component is attached to. A component is always attached to a game object.
tag	The tag of this game object.
transform	The Transform attached to this GameObject.
runInEditMode	Allow a specific instance of a MonoBehaviour to run in edit mode (only available in the editor).
useGUILayout	Disabling this lets you skip the GUI layout phase.
hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.

Public Methods

BroadcastMessage	Calls the method named methodName on every MonoBehaviour in this game object or any of its children.
CompareTag	Is this game object tagged with tag ?
GetComponent	Returns the component of Type type if the game object has one attached, null if it doesn't.
GetComponentInChildren	Returns the component of Type type in the GameObject or any of its children using depth first search.
GetComponentInParent	Returns the component of Type type in the GameObject or any of its parents.
GetComponents	Returns all components of Type type in the GameObject.
GetComponentsInChildren	Returns all components of Type type in the GameObject or any of its children.
GetComponentsInParent	Returns all components of Type type in the GameObject or any of its parents.
SendMessage	Calls the method named methodName on every MonoBehaviour in this game

	object.
SendMessageUpwards	Calls the method named methodName on every MonoBehaviour in this game object and on every ancestor of the behaviour.
CancelInvoke	Cancels all Invoke calls on this MonoBehaviour.
Invoke	Invokes the method methodName in time seconds.
InvokeRepeating	Invokes the method methodName in time seconds, then repeatedly every repeatRate seconds.
IsInvoking	Is any invoke on methodName pending?
StartCoroutine	Starts a coroutine.
StopAllCoroutines	Stops all coroutines running on this behaviour.
StopCoroutine	Stops the first coroutine named methodName, or the coroutine stored in routine running on this behaviour.
GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.
IsActive	Returns true if the GameObject and the Component are active.
IsDestroyed	Returns true if the native representation of the behaviour has been destroyed.

Protected Methods

Awake	See MonoBehaviour.Awake.
OnBeforeTransformParentChanged	See MonoBehaviour.OnBeforeTransformParentChanged.
OnCanvasGroupChanged	See MonoBehaviour.OnCanvasGroupChanged.
OnCanvasHierarchyChanged	Called when the state of the parent Canvas is changed.
OnDestroy	See MonoBehaviour.OnDestroy.
OnDidApplyAnimationProperties	See LayoutGroup.OnDidApplyAnimationProperties.
OnEnable	See MonoBehaviour.OnEnable.
OnRectTransformDimensionsChange	This callback is called if an associated RectTransform has its dimensions changed.
OnTransformParentChanged	See MonoBehaviour.OnRectTransformParentChanged.
OnValidate	See MonoBehaviour.OnValidate.
Reset	See MonoBehaviour.Reset.

Start	See MonoBehaviour.Start.

Static Methods

print	Logs message to the Unity Console (identical to Debug.Log).
Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Messages

Awake	Awake is called when the script instance is being loaded.
FixedUpdate	This function is called every fixed framerate frame, if the MonoBehaviour is enabled.
LateUpdate	LateUpdate is called every frame, if the Behaviour is enabled.
OnAnimatorIK	Callback for setting up animation IK (inverse kinematics).
OnAnimatorMove	Callback for processing animation movements for modifying root motion.
OnApplicationFocus	Sent to all GameObjects when the player gets or loses focus.
OnApplicationPause	Sent to all GameObjects when the application pauses.
OnApplicationQuit	Sent to all game objects before the application quits.
OnAudioFilterRead	If OnAudioFilterRead is implemented, Unity will insert a custom filter into the audio DSP chain.

OnBecameInvisible	OnBecameInvisible is called when the renderer is no longer visible by any camera.
OnBecameVisible	OnBecameVisible is called when the renderer became visible by any camera.
OnCollisionEnter	OnCollisionEnter is called when this collider/rigidbody has begun touching another rigidbody/collider.
OnCollisionEnter2D	Sent when an incoming collider makes contact with this object's collider (2D physics only).
OnCollisionExit	OnCollisionExit is called when this collider/rigidbody has stopped touching another rigidbody/collider.
OnCollisionExit2D	Sent when a collider on another object stops touching this object's collider (2D physics only).
OnCollisionStay	:ref::OnCollisionStay is called once per frame for every collider/rigidbody that is touching rigidbody/collider.
OnCollisionStay2D	Sent each frame where a collider on another object is touching this object's collider (2D physics only).
OnConnectedToServer	Called on the client when you have successfully connected to a server.
OnControllerColliderHit	OnControllerColliderHit is called when the controller hits a collider while performing a Move.
OnDestroy	Destroying the attached Behaviour will result in the game or Scene receiving OnDestroy.
OnDisconnectedFromServer	Called on the client when the connection was lost or you disconnected from the server.
OnDrawGizmos	Implement OnDrawGizmos if you want to draw gizmos that are also pickable and always drawn.
OnDrawGizmosSelected	Implement OnDrawGizmosSelected to draw a gizmo if the object is selected.
OnEnable	This function is called when the object becomes enabled and active.
OnFailedToConnect	Called on the client when a connection attempt fails for some reason.
OnFailedToConnectToMasterServer	Called on clients or servers when there is a problem connecting to the MasterServer.
OnGUI	OnGUI is called for rendering and handling GUI events.
OnJointBreak	Called when a joint attached to the same game object broke.
OnJointBreak2D	Called when a Joint2D attached to the same game object breaks.

OnMasterServerEvent	Called on clients or servers when reporting events from the MasterServer.
OnMouseDown	OnMouseDown is called when the user has pressed the mouse button while over the GUIElement or Collider.
OnMouseDrag	OnMouseDrag is called when the user has clicked on a GUIElement or Collider and is still holding down the mouse.
OnMouseEnter	Called when the mouse enters the GUIElement or Collider.
OnMouseExit	Called when the mouse is not any longer over the GUIElement or Collider.
OnMouseOver	Called every frame while the mouse is over the GUIElement or Collider.
OnMouseUp	OnMouseUp is called when the user has released the mouse button.
OnMouseUpAsButton	OnMouseUpAsButton is only called when the mouse is released over the same GUIElement or Collider as it was pressed.
OnNetworkInstantiate	Called on objects which have been network instantiated with Network.Instantiate.
OnParticleCollision	OnParticleCollision is called when a particle hits a Collider.
OnParticleSystemStopped	OnParticleSystemStopped is called when all particles in the system have died, and no new particles will be born. New particles cease to be created either after Stop is called, or when the duration property of a non-looping system has been exceeded.
OnParticleTrigger	OnParticleTrigger is called when any particles in a particle system meet the conditions in the trigger module.
OnPlayerConnected	Called on the server whenever a new player has successfully connected.
OnPlayerDisconnected	Called on the server whenever a player disconnected from the server.
OnPostRender	OnPostRender is called after a camera finished rendering the Scene.
OnPreCull	OnPreCull is called before a camera culls the Scene.
OnPreRender	OnPreRender is called before a camera starts rendering the Scene.
OnRenderImage	OnRenderImage is called after all rendering is complete to render image.
OnRenderObject	OnRenderObject is called after camera has rendered the Scene.
OnSerializeNetworkView	Used to customize synchronization of variables in a script watched by a network view.

OnServerInitialized	Called on the server whenever a Network.InitializeServer was invoked and has completed.
OnTransformChildrenChanged	This function is called when the list of children of the transform of the GameObject has changed.
OnTransformParentChanged	This function is called when the parent property of the transform of the GameObject has changed.
OnTriggerEnter	OnTriggerEnter is called when the Collider other enters the trigger.
OnTriggerEnter2D	Sent when another object enters a trigger collider attached to this object (2D physics only).
OnTriggerExit	OnTriggerExit is called when the Collider other has stopped touching the trigger.
OnTriggerExit2D	Sent when another object leaves a trigger collider attached to this object (2D physics only).
OnTriggerStay	OnTriggerStay is called once per physics update for every Collider other that is touching the trigger.
OnTriggerStay2D	Sent each frame where another object is within a trigger collider attached to this object (2D physics only).
OnValidate	This function is called when the script is loaded or a value is changed in the inspector (Called in the editor only).
OnWillRenderObject	OnWillRenderObject is called for each camera if the object is visible and not a UI element.
Reset	Reset to default values.
Start	Start is called on the frame when a script is enabled just before any of the Update methods are called the first time.
Update	Update is called every frame, if the MonoBehaviour is enabled.

Obsolete

Description

Base class for effects that modify the the generated Vertex Buffers.

Implements interfaces:IEventSystemHandler, IPointerClickHandler, ISubmitHandler

Description

A standard button that can be clicked in order to trigger an event.

See Selectable for selection states.

Properties

onClick	UnityEvent that is triggered when the Button is pressed.

Public Methods

OnPointerClick	Registered IPointerClickHandler callback.
OnSubmit	Registered ISubmitHandler callback.

Inherited Members

Static Properties

allSelectables	List of all the selectable objects currently active in the Scene.

Properties

enabled	Enabled Behaviours are Updated, disabled Behaviours are not.
isActiveAndEnabled	Has the Behaviour had active and enabled called?
gameObject	The game object this component is attached to. A component is always attached to a game object.
tag	The tag of this game object.
transform	The Transform attached to this GameObject.
runInEditMode	Allow a specific instance of a MonoBehaviour to run in edit mode (only available in the editor).
useGUILayout	Disabling this lets you skip the GUI layout phase.
hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.

animationTriggers	The AnimationTriggers for this selectable object.
animator	Convenience function to get the Animator component on the GameObject.
colors	The ColorBlock for this selectable object.
image	Convenience function that converts the referenced Graphic to a Image, if possible.
interactable	Use to enable or disable the ability to select a selectable UI element (for example, a Button).
navigation	The Navigation setting for this selectable object.
spriteState	The SpriteState for this selectable object.
targetGraphic	Graphic that will be transitioned upon.
transition	The type of transition that will be applied to the targetGraphic when the state changes.

Public Methods

BroadcastMessage	Calls the method named methodName on every MonoBehaviour in this game object or any of its children.
CompareTag	Is this game object tagged with tag ?
GetComponent	Returns the component of Type type if the game object has one attached, null if it doesn't.
GetComponentInChildren	Returns the component of Type type in the GameObject or any of its children using depth first search.
GetComponentInParent	Returns the component of Type type in the GameObject or any of its parents.
GetComponents	Returns all components of Type type in the GameObject.
GetComponentsInChildren	Returns all components of Type type in the GameObject or any of its children.
GetComponentsInParent	Returns all components of Type type in the GameObject or any of its parents.
SendMessage	Calls the method named methodName on every MonoBehaviour in this game object.
SendMessageUpwards	Calls the method named methodName on every MonoBehaviour in this game object and on every ancestor of the behaviour.
CancelInvoke	Cancels all Invoke calls on this MonoBehaviour.
Invoke	Invokes the method methodName in time seconds.
InvokeRepeating	Invokes the method methodName in time seconds, then repeatedly every repeatRate seconds.

IsInvoking	Is any invoke on methodName pending?
StartCoroutine	Starts a coroutine.
StopAllCoroutines	Stops all coroutines running on this behaviour.
StopCoroutine	Stops the first coroutine named methodName, or the coroutine stored in routine running on this behaviour.
GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.
FindSelectable	Finds the selectable object next to this one.
FindSelectableOnDown	Find the selectable object below this one.
FindSelectableOnLeft	Find the selectable object to the left of this one.
FindSelectableOnRight	Find the selectable object to the right of this one.
FindSelectableOnUp	Find the selectable object above this one.
IsInteractable	UI.Selectable.IsInteractable.
OnDeselect	Unset selection and transition to appropriate state.
OnMove	Determine in which of the 4 move directions the next selectable object should be found.
OnPointerDown	Evaluate current state and transition to pressed state.
OnPointerEnter	Evaluate current state and transition to appropriate state.
OnPointerExit	Evaluate current state and transition to normal state.
OnPointerUp	Evaluate eventData and transition to appropriate state.
OnSelect	Set selection and transition to appropriate state.
Select	Selects this Selectable.
IsActive	Returns true if the GameObject and the Component are active.
IsDestroyed	Returns true if the native representation of the behaviour has been destroyed.

Protected Methods

DoStateTransition	Transition the Selectable to the entered state.
InstantClearState	Clear any internal state from the Selectable (used when disabling).
IsHighlighted	Returns whether the selectable is currently 'highlighted' or not.

IsPressed	Whether the current selectable is being pressed.
OnDisable	See MonoBehaviour.OnDisable.
UpdateSelectionState	Internally update the selection state of the Selectable.
Awake	See MonoBehaviour.Awake.
OnBeforeTransformParentChanged	See MonoBehaviour.OnBeforeTransformParentChanged.
OnCanvasGroupChanged	See MonoBehaviour.OnCanvasGroupChanged.
OnCanvasHierarchyChanged	Called when the state of the parent Canvas is changed.
OnDestroy	See MonoBehaviour.OnDestroy.
OnDidApplyAnimationProperties	See LayoutGroup.OnDidApplyAnimationProperties.
OnDisable	See MonoBehaviour.OnDisable.
OnEnable	See MonoBehaviour.OnEnable.
OnRectTransformDimensionsChange	This callback is called if an associated RectTransform has its dimensions changed.
OnTransformParentChanged	See MonoBehaviour.OnRectTransformParentChanged.
OnValidate	See MonoBehaviour.OnValidate.
Reset	See MonoBehaviour.Reset.
Start	See MonoBehaviour.Start.

Static Methods

print	Logs message to the Unity Console (identical to Debug.Log).
Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Messages

Awake	Awake is called when the script instance is being loaded.
FixedUpdate	This function is called every fixed framerate frame, if the MonoBehaviour is enabled.
LateUpdate	LateUpdate is called every frame, if the Behaviour is enabled.
OnAnimatorIK	Callback for setting up animation IK (inverse kinematics).
OnAnimatorMove	Callback for processing animation movements for modifying root motion.
OnApplicationFocus	Sent to all GameObjects when the player gets or loses focus.
OnApplicationPause	Sent to all GameObjects when the application pauses.
OnApplicationQuit	Sent to all game objects before the application quits.
OnAudioFilterRead	If OnAudioFilterRead is implemented, Unity will insert a custom filter into the audio DSP chain.
OnBecameInvisible	OnBecameInvisible is called when the renderer is no longer visible by any camera.
OnBecameVisible	OnBecameVisible is called when the renderer became visible by any camera.
OnCollisionEnter	OnCollisionEnter is called when this collider/rigidbody has begun touching another rigidbody/collider.
OnCollisionEnter2D	Sent when an incoming collider makes contact with this object's collider (2D physics only).
OnCollisionExit	OnCollisionExit is called when this collider/rigidbody has stopped touching another rigidbody/collider.
OnCollisionExit2D	Sent when a collider on another object stops touching this object's collider (2D physics only).
OnCollisionStay	:ref::OnCollisionStay is called once per frame for every collider/rigidbody that is touching rigidbody/collider.
OnCollisionStay2D	Sent each frame where a collider on another object is touching this object's collider (2D physics only).

OnConnectedToServer	Called on the client when you have successfully connected to a server.
OnControllerColliderHit	OnControllerColliderHit is called when the controller hits a collider while performing a Move.
OnDestroy	Destroying the attached Behaviour will result in the game or Scene receiving OnDestroy.
OnDisable	This function is called when the behaviour becomes disabled.
OnDisconnectedFromServer	Called on the client when the connection was lost or you disconnected from the server.
OnDrawGizmos	Implement OnDrawGizmos if you want to draw gizmos that are also pickable and always drawn.
OnDrawGizmosSelected	Implement OnDrawGizmosSelected to draw a gizmo if the object is selected.
OnEnable	This function is called when the object becomes enabled and active.
OnFailedToConnect	Called on the client when a connection attempt fails for some reason.
OnFailedToConnectToMasterServer	Called on clients or servers when there is a problem connecting to the MasterServer.
OnGUI	OnGUI is called for rendering and handling GUI events.
OnJointBreak	Called when a joint attached to the same game object broke.
OnJointBreak2D	Called when a Joint2D attached to the same game object breaks.
OnMasterServerEvent	Called on clients or servers when reporting events from the MasterServer.
OnMouseDown	OnMouseDown is called when the user has pressed the mouse button while over the GUIElement or Collider.
OnMouseDrag	OnMouseDrag is called when the user has clicked on a GUIElement or Collider and is still holding down the mouse.
OnMouseEnter	Called when the mouse enters the GUIElement or Collider.
OnMouseExit	Called when the mouse is not any longer over the GUIElement or Collider.
OnMouseOver	Called every frame while the mouse is over the GUIElement or Collider.
OnMouseUp	OnMouseUp is called when the user has released the mouse button.
OnMouseUpAsButton	OnMouseUpAsButton is only called when the mouse is released over the same GUIElement or Collider as it was pressed.

OnNetworkInstantiate	Called on objects which have been network instantiated with Network.Instantiate.
OnParticleCollision	OnParticleCollision is called when a particle hits a Collider.
OnParticleSystemStopped	OnParticleSystemStopped is called when all particles in the system have died, and no new particles will be born. New particles cease to be created either after Stop is called, or when the duration property of a non-looping system has been exceeded.
OnParticleTrigger	OnParticleTrigger is called when any particles in a particle system meet the conditions in the trigger module.
OnPlayerConnected	Called on the server whenever a new player has successfully connected.
OnPlayerDisconnected	Called on the server whenever a player disconnected from the server.
OnPostRender	OnPostRender is called after a camera finished rendering the Scene.
OnPreCull	OnPreCull is called before a camera culls the Scene.
OnPreRender	OnPreRender is called before a camera starts rendering the Scene.
OnRenderImage	OnRenderImage is called after all rendering is complete to render image.
OnRenderObject	OnRenderObject is called after camera has rendered the Scene.
OnSerializeNetworkView	Used to customize synchronization of variables in a script watched by a network view.
OnServerInitialized	Called on the server whenever a Network.InitializeServer was invoked and has completed.
OnTransformChildrenChanged	This function is called when the list of children of the transform of the GameObject has changed.
OnTransformParentChanged	This function is called when the parent property of the transform of the GameObject has changed.
OnTriggerEnter	OnTriggerEnter is called when the Collider other enters the trigger.
OnTriggerEnter2D	Sent when another object enters a trigger collider attached to this object (2D physics only).
OnTriggerExit	OnTriggerExit is called when the Collider other has stopped touching the trigger.
OnTriggerExit2D	Sent when another object leaves a trigger collider attached to this object (2D physics only).
OnTriggerStay	OnTriggerStay is called once per physics update for every Collider other that is touching the trigger.

OnTriggerStay2D	Sent each frame where another object is within a trigger collider attached to this object (2D physics only).
OnValidate	This function is called when the script is loaded or a value is changed in the inspector (Called in the editor only).
OnWillRenderObject	OnWillRenderObject is called for each camera if the object is visible and not a UI element.
Reset	Reset to default values.
Start	Start is called on the frame when a script is enabled just before any of the Update methods are called the first time.
Update	Update is called every frame, if the MonoBehaviour is enabled.

Description

Function definition for a button click event.

Inherited Members

Public Methods

AddListener	Add a non persistent listener to the UnityEvent.
Invoke	Invoke all registered callbacks (runtime and persistent).
RemoveListener	Remove a non persistent listener from the UnityEvent.
GetPersistentEventCount	Get the number of registered persistent listeners.
GetPersistentMethodName	Get the target method name of the listener at index index.
GetPersistentTarget	Get the target component of the listener at index index.
RemoveAllListeners	Remove all non-persisent (ie created from script) listeners from the event.
SetPersistentListenerState	Modify the execution state of a persistent listener.

Static Methods

GetValidMethodInfo	Given an object, function name, and a list of argument types; find the method that matches.

Description

The Canvas Scaler component is used for controlling the overall scale and pixel density of UI elements in the Canvas. This scaling affects everything under the Canvas, including font sizes and image borders.

For a Canvas set to 'Screen Space - Overlay' or 'Screen Space - Camera', the Canvas Scaler UI Scale Mode can be set to Constant Pixel Size, Scale With Screen Size, or Constant Physical Size.

Using the Constant Pixel Size mode, positions and sizes of UI elements are specified in pixels on the screen. This is also the default functionality of the Canvas when no Canvas Scaler is attached. However, With the Scale Factor setting in the Canvas Scaler, a constant scaling can be applied to all UI elements in the Canvas.

Using the Scale With Screen Size mode, positions and sizes can be specified according to the pixels of a specified reference resolution. If the current screen resolution is larger than the reference resolution, the Canvas will keep having only the resolution of the reference resolution, but will scale up in order to fit the screen. If the current screen resolution is smaller than the reference resolution, the Canvas will similarly be scaled down to fit. If the current screen resolution has a different aspect ratio than the reference resolution, scaling each axis individually to fit the screen would result in non-uniform scaling, which is generally undesirable. Instead of this, the ReferenceResolution component will make the Canvas resolution deviate from the reference resolution in order to respect the aspect ratio of the screen. It is possible to control how this deviation should behave using the screenMatchMode setting.

Using the Constant Physical Size mode, positions and sizes of UI elements are specified in physical units, such as millimeters, points, or picas. This mode relies on the device reporting its screen DPI correctly. You can specify a fallback DPI to use for devices that do not report a DPI.

For a Canvas set to 'World Space' the Canvas Scaler can be used to control the pixel density of UI elements in the Canvas.

See Also: Canvas.

Properties

defaultSpriteDPI	The pixels per inch to use for sprites that have a 'Pixels Per Unit' setting that matches the 'Reference Pixels Per Unit' setting.
dynamicPixelsPerUnit	The amount of pixels per unit to use for dynamically created bitmaps in the UI, such as Text.
fallbackScreenDPI	The DPI to assume if the screen DPI is not known.
matchWidthOrHeight	Setting to scale the Canvas to match the width or height of the reference resolution, or a combination.
physicalUnit	The physical unit to specify positions and sizes in.

referencePixelsPerUnit	If a sprite has this 'Pixels Per Unit' setting, then one pixel in the sprite will cover one unit in the UI.
referenceResolution	The resolution the UI layout is designed for.
scaleFactor	Scales all UI elements in the Canvas by this factor.
screenMatchMode	A mode used to scale the canvas area if the aspect ratio of the current resolution doesn't fit the reference resolution.
uiScaleMode	Determines how UI elements in the Canvas are scaled.

Protected Methods

Handle	Method that handles calculations of canvas scaling.
HandleConstantPhysicalSize	Handles canvas scaling for a constant physical size.
HandleConstantPixelSize	Handles canvas scaling for a constant pixel size.
HandleScaleWithScreenSize	Handles canvas scaling that scales with the screen size.
HandleWorldCanvas	Handles canvas scaling for world canvas.
OnDisable	See MonoBehaviour.OnDisable.
SetReferencePixelsPerUnit	Sets the referencePixelsPerUnit on the Canvas.
SetScaleFactor	Sets the scale factor on the canvas.
Update	Handles per-frame checking if the canvas scaling needs to be updated.

Inherited Members

Properties

enabled	Enabled Behaviours are Updated, disabled Behaviours are not.
isActiveAndEnabled	Has the Behaviour had active and enabled called?
gameObject	The game object this component is attached to. A component is always attached to a game object.
tag	The tag of this game object.
transform	The Transform attached to this GameObject.
runInEditMode	Allow a specific instance of a MonoBehaviour to run in edit mode (only available in the editor).
useGUILayout	Disabling this lets you skip the GUI layout phase.

hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.

Public Methods

BroadcastMessage	Calls the method named methodName on every MonoBehaviour in this game object or any of its children.
CompareTag	Is this game object tagged with tag ?
GetComponent	Returns the component of Type type if the game object has one attached, null if it doesn't.
GetComponentInChildren	Returns the component of Type type in the GameObject or any of its children using depth first search.
GetComponentInParent	Returns the component of Type type in the GameObject or any of its parents.
GetComponents	Returns all components of Type type in the GameObject.
GetComponentsInChildren	Returns all components of Type type in the GameObject or any of its children.
GetComponentsInParent	Returns all components of Type type in the GameObject or any of its parents.
SendMessage	Calls the method named methodName on every MonoBehaviour in this game object.
SendMessageUpwards	Calls the method named methodName on every MonoBehaviour in this game object and on every ancestor of the behaviour.
CancelInvoke	Cancels all Invoke calls on this MonoBehaviour.
Invoke	Invokes the method methodName in time seconds.
InvokeRepeating	Invokes the method methodName in time seconds, then repeatedly every repeatRate seconds.
IsInvoking	Is any invoke on methodName pending?
StartCoroutine	Starts a coroutine.
StopAllCoroutines	Stops all coroutines running on this behaviour.
StopCoroutine	Stops the first coroutine named methodName, or the coroutine stored in routine running on this behaviour.
GetInstanceID	Returns the instance id of the object.
ToString	Returns the name of the GameObject.
IsActive	Returns true if the GameObject and the Component are active.

IsDestroyed	Returns true if the native representation of the behaviour has been destroyed.

Protected Methods

Awake	See MonoBehaviour.Awake.
OnBeforeTransformParentChanged	See MonoBehaviour.OnBeforeTransformParentChanged.
OnCanvasGroupChanged	See MonoBehaviour.OnCanvasGroupChanged.
OnCanvasHierarchyChanged	Called when the state of the parent Canvas is changed.
OnDestroy	See MonoBehaviour.OnDestroy.
OnDidApplyAnimationProperties	See LayoutGroup.OnDidApplyAnimationProperties.
OnEnable	See MonoBehaviour.OnEnable.
OnRectTransformDimensionsChange	This callback is called if an associated RectTransform has its dimensions changed.
OnTransformParentChanged	See MonoBehaviour.OnRectTransformParentChanged.
OnValidate	See MonoBehaviour.OnValidate.
Reset	See MonoBehaviour.Reset.
Start	See MonoBehaviour.Start.

Static Methods

print	Logs message to the Unity Console (identical to Debug.Log).
Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.

Operators

bool	Does the object exist?

operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Messages

Awake	Awake is called when the script instance is being loaded.
FixedUpdate	This function is called every fixed framerate frame, if the MonoBehaviour is enabled.
LateUpdate	LateUpdate is called every frame, if the Behaviour is enabled.
OnAnimatorIK	Callback for setting up animation IK (inverse kinematics).
OnAnimatorMove	Callback for processing animation movements for modifying root motion.
OnApplicationFocus	Sent to all GameObjects when the player gets or loses focus.
OnApplicationPause	Sent to all GameObjects when the application pauses.
OnApplicationQuit	Sent to all game objects before the application quits.
OnAudioFilterRead	If OnAudioFilterRead is implemented, Unity will insert a custom filter into the audio DSP chain.
OnBecameInvisible	OnBecameInvisible is called when the renderer is no longer visible by any camera.
OnBecameVisible	OnBecameVisible is called when the renderer became visible by any camera.
OnCollisionEnter	OnCollisionEnter is called when this collider/rigidbody has begun touching another rigidbody/collider.
OnCollisionEnter2D	Sent when an incoming collider makes contact with this object's collider (2D physics only).
OnCollisionExit	OnCollisionExit is called when this collider/rigidbody has stopped touching another rigidbody/collider.
OnCollisionExit2D	Sent when a collider on another object stops touching this object's collider (2D physics only).
OnCollisionStay	:ref::OnCollisionStay is called once per frame for every collider/rigidbody that is touching rigidbody/collider.
OnCollisionStay2D	Sent each frame where a collider on another object is touching this object's collider (2D physics only).
OnConnectedToServer	Called on the client when you have successfully connected to a server.

OnControllerColliderHit	OnControllerColliderHit is called when the controller hits a collider while performing a Move.
OnDestroy	Destroying the attached Behaviour will result in the game or Scene receiving OnDestroy.
OnDisconnectedFromServer	Called on the client when the connection was lost or you disconnected from the server.
OnDrawGizmos	Implement OnDrawGizmos if you want to draw gizmos that are also pickable and always drawn.
OnDrawGizmosSelected	Implement OnDrawGizmosSelected to draw a gizmo if the object is selected.
OnEnable	This function is called when the object becomes enabled and active.
OnFailedToConnect	Called on the client when a connection attempt fails for some reason.
OnFailedToConnectToMasterServer	Called on clients or servers when there is a problem connecting to the MasterServer.
OnGUI	OnGUI is called for rendering and handling GUI events.
OnJointBreak	Called when a joint attached to the same game object broke.
OnJointBreak2D	Called when a Joint2D attached to the same game object breaks.
OnMasterServerEvent	Called on clients or servers when reporting events from the MasterServer.
OnMouseDown	OnMouseDown is called when the user has pressed the mouse button while over the GUIElement or Collider.
OnMouseDrag	OnMouseDrag is called when the user has clicked on a GUIElement or Collider and is still holding down the mouse.
OnMouseEnter	Called when the mouse enters the GUIElement or Collider.
OnMouseExit	Called when the mouse is not any longer over the GUIElement or Collider.
OnMouseOver	Called every frame while the mouse is over the GUIElement or Collider.
OnMouseUp	OnMouseUp is called when the user has released the mouse button.
OnMouseUpAsButton	OnMouseUpAsButton is only called when the mouse is released over the same GUIElement or Collider as it was pressed.
OnNetworkInstantiate	Called on objects which have been network instantiated with Network.Instantiate.

OnParticleCollision	OnParticleCollision is called when a particle hits a Collider.
OnParticleSystemStopped	OnParticleSystemStopped is called when all particles in the system have died, and no new particles will be born. New particles cease to be created either after Stop is called, or when the duration property of a non-looping system has been exceeded.
OnParticleTrigger	OnParticleTrigger is called when any particles in a particle system meet the conditions in the trigger module.
OnPlayerConnected	Called on the server whenever a new player has successfully connected.
OnPlayerDisconnected	Called on the server whenever a player disconnected from the server.
OnPostRender	OnPostRender is called after a camera finished rendering the Scene.
OnPreCull	OnPreCull is called before a camera culls the Scene.
OnPreRender	OnPreRender is called before a camera starts rendering the Scene.
OnRenderImage	OnRenderImage is called after all rendering is complete to render image.
OnRenderObject	OnRenderObject is called after camera has rendered the Scene.
OnSerializeNetworkView	Used to customize synchronization of variables in a script watched by a network view.
OnServerInitialized	Called on the server whenever a Network.InitializeServer was invoked and has completed.
OnTransformChildrenChanged	This function is called when the list of children of the transform of the GameObject has changed.
OnTransformParentChanged	This function is called when the parent property of the transform of the GameObject has changed.
OnTriggerEnter	OnTriggerEnter is called when the Collider other enters the trigger.
OnTriggerEnter2D	Sent when another object enters a trigger collider attached to this object (2D physics only).
OnTriggerExit	OnTriggerExit is called when the Collider other has stopped touching the trigger.
OnTriggerExit2D	Sent when another object leaves a trigger collider attached to this object (2D physics only).
OnTriggerStay	OnTriggerStay is called once per physics update for every Collider other that is touching the trigger.
OnTriggerStay2D	Sent each frame where another object is within a trigger collider attached to this object (2D physics only).

OnValidate	This function is called when the script is loaded or a value is changed in the inspector (Called in the editor only).
OnWillRenderObject	OnWillRenderObject is called for each camera if the object is visible and not a UI element.
Reset	Reset to default values.
Start	Start is called on the frame when a script is enabled just before any of the Update methods are called the first time.

Description

Determines how UI elements in the Canvas are scaled.

Properties

ConstantPixelSize	Using the Constant Pixel Size mode, positions and sizes of UI elements are specified in pixels on the screen.
ScaleWithScreenSize	Using the Scale With Screen Size mode, positions and sizes can be specified according to the pixels of a specified reference resolution. If the current screen resolution is larger than the reference resolution, the Canvas will keep having only the resolution of the reference resolution, but will scale up in order to fit the screen. If the current screen resolution is smaller than the reference resolution, the Canvas will similarly be scaled down to fit.
ConstantPhysicalSize	Using the Constant Physical Size mode, positions and sizes of UI elements are specified in physical units, such as millimeters, points, or picas.

Description

Scale the canvas area with the width as reference, the height as reference, or something in between.

Properties

MatchWidthOrHeight	Scale the canvas area with the width as reference, the height as reference, or something in between.
Expand	Expand the canvas area either horizontally or vertically, so the size of the canvas will never be smaller than the reference.
Shrink	Crop the canvas area either horizontally or vertically, so the size of the canvas will never be larger than the reference.

Description

Settings used to specify a physical unit.

Properties

Centimeters	Use centimeters. A centimeter is 1/100 of a meter.
Millimeters	Use millimeters. A millimeter is 1/10 of a centimeter, and 1/1000 of a meter.
Inches	Use inches.
Points	Use points. One point is 1/12 of a pica, and 1/72 of an inch.
Picas	Use picas. One pica is 1/6 of an inch.

Description

A place where CanvasElements can register themselves for rebuilding.

Static Properties

instance	Get the singleton registry.

Static Methods

IsRebuildingGraphics	Are graphics being rebuild.
IsRebuildingLayout	Is layout being rebuilt?
RegisterCanvasElementForGraphicRebuild	Rebuild the graphics of the given element.
RegisterCanvasElementForLayoutRebuild	Rebuild the layout of the given element.
TryRegisterCanvasElementForGraphicRebuild	Rebuild the layout of the given element.
TryRegisterCanvasElementForLayoutRebuild	Was the element scheduled.
UnRegisterCanvasElementForRebuild	Remove the given element from rebuild.

Description

Registry class to keep track of all IClippers that exist in the Scene.

This is used during the CanvasUpdate loop to cull clippable elements. The clipping is called after layout, but before Graphic update.

Static Properties

instance	Singleton instance.

Public Methods

Cull	Perform the clipping on all registered IClipper.

Static Methods

Register	Register an IClipper.
Unregister	Unregister an IClipper.

Description

Utility class to help when clipping using IClipper.

Static Methods

FindCullAndClipWorldRect	Find the Rect to use for clipping.

Description

Structure to store the state of a color transition on a Selectable.

Static Properties

defaultColorBlock	Simple getter for the default ColorBlock.

Properties

colorMultiplier	Multiplier applied to colors (allows brightening greater then base color).
disabledColor	Disabled Color.
fadeDuration	How long a color transition should take.
highlightedColor	Highlighted Color.
normalColor	Normal Color.
pressedColor	Pressed Color.

Implements interfaces:ILayoutController, ILayoutSelfController

Description

Resizes a RectTransform to fit the size of its content.

The ContentSizeFitter can be used on GameObjects that have one or more ILayoutElement components, such as Text, Image, HorizontalLayoutGroup, VerticalLayoutGroup, and GridLayoutGroup.

See Also: Auto Layout.

Properties

horizontalFit	The fit mode to use to determine the width.
verticalFit	The fit mode to use to determine the height.

Public Methods

SetLayoutHorizontal	Method called by the layout system.
SetLayoutVertical	Method called by the layout system.

Protected Methods

OnDisable	See MonoBehaviour.OnDisable.
SetDirty	Mark the ContentSizeFitter as dirty.

Inherited Members

Properties

enabled	Enabled Behaviours are Updated, disabled Behaviours are not.
isActiveAndEnabled	Has the Behaviour had active and enabled called?
gameObject	The game object this component is attached to. A component is always attached to a game object.
tag	The tag of this game object.
transform	The Transform attached to this GameObject.
runInEditMode	Allow a specific instance of a MonoBehaviour to run in edit mode (only available in the

	editor).
useGUILayout	Disabling this lets you skip the GUI layout phase.
hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.

Public Methods

BroadcastMessage	Calls the method named methodName on every MonoBehaviour in this game object or any of its children.
CompareTag	Is this game object tagged with tag ?
GetComponent	Returns the component of Type type if the game object has one attached, null if it doesn't.
GetComponentInChildren	Returns the component of Type type in the GameObject or any of its children using depth first search.
GetComponentInParent	Returns the component of Type type in the GameObject or any of its parents.
GetComponents	Returns all components of Type type in the GameObject.
GetComponentsInChildren	Returns all components of Type type in the GameObject or any of its children.
GetComponentsInParent	Returns all components of Type type in the GameObject or any of its parents.
SendMessage	Calls the method named methodName on every MonoBehaviour in this game object.
SendMessageUpwards	Calls the method named methodName on every MonoBehaviour in this game object and on every ancestor of the behaviour.
CancelInvoke	Cancels all Invoke calls on this MonoBehaviour.
Invoke	Invokes the method methodName in time seconds.
InvokeRepeating	Invokes the method methodName in time seconds, then repeatedly every repeatRate seconds.
IsInvoking	Is any invoke on methodName pending?
StartCoroutine	Starts a coroutine.
StopAllCoroutines	Stops all coroutines running on this behaviour.
StopCoroutine	Stops the first coroutine named methodName, or the coroutine stored in routine running on this behaviour.
GetInstanceID	Returns the instance id of the object.

ToString	Returns the name of the GameObject.
IsActive	Returns true if the GameObject and the Component are active.
IsDestroyed	Returns true if the native representation of the behaviour has been destroyed.

Protected Methods

Awake	See MonoBehaviour.Awake.
OnBeforeTransformParentChanged	See MonoBehaviour.OnBeforeTransformParentChanged.
OnCanvasGroupChanged	See MonoBehaviour.OnCanvasGroupChanged.
OnCanvasHierarchyChanged	Called when the state of the parent Canvas is changed.
OnDestroy	See MonoBehaviour.OnDestroy.
OnDidApplyAnimationProperties	See LayoutGroup.OnDidApplyAnimationProperties.
OnEnable	See MonoBehaviour.OnEnable.
OnRectTransformDimensionsChange	This callback is called if an associated RectTransform has its dimensions changed.
OnTransformParentChanged	See MonoBehaviour.OnRectTransformParentChanged.
OnValidate	See MonoBehaviour.OnValidate.
Reset	See MonoBehaviour.Reset.
Start	See MonoBehaviour.Start.

Static Methods

print	Logs message to the Unity Console (identical to Debug.Log).
Destroy	Removes a gameobject, component or asset.
DestroyImmediate	Destroys the object obj immediately. You are strongly recommended to use Destroy instead.
DontDestroyOnLoad	Makes the object target not be destroyed automatically when loading a new Scene.
FindObjectOfType	Returns the first active loaded object of Type type.
FindObjectsOfType	Returns a list of all active loaded objects of Type type.
Instantiate	Clones the object original and returns the clone.

Operators

bool	Does the object exist?
operator !=	Compares if two objects refer to a different object.
operator ==	Compares two object references to see if they refer to the same object.

Messages

Awake	Awake is called when the script instance is being loaded.
FixedUpdate	This function is called every fixed framerate frame, if the MonoBehaviour is enabled.
LateUpdate	LateUpdate is called every frame, if the Behaviour is enabled.
OnAnimatorIK	Callback for setting up animation IK (inverse kinematics).
OnAnimatorMove	Callback for processing animation movements for modifying root motion.
OnApplicationFocus	Sent to all GameObjects when the player gets or loses focus.
OnApplicationPause	Sent to all GameObjects when the application pauses.
OnApplicationQuit	Sent to all game objects before the application quits.
OnAudioFilterRead	If OnAudioFilterRead is implemented, Unity will insert a custom filter into the audio DSP chain.
OnBecameInvisible	OnBecameInvisible is called when the renderer is no longer visible by any camera.
OnBecameVisible	OnBecameVisible is called when the renderer became visible by any camera.
OnCollisionEnter	OnCollisionEnter is called when this collider/rigidbody has begun touching another rigidbody/collider.
OnCollisionEnter2D	Sent when an incoming collider makes contact with this object's collider (2D physics only).
OnCollisionExit	OnCollisionExit is called when this collider/rigidbody has stopped touching another rigidbody/collider.
OnCollisionExit2D	Sent when a collider on another object stops touching this object's collider (2D physics only).
OnCollisionStay	:ref::OnCollisionStay is called once per frame for every collider/rigidbody that is touching rigidbody/collider.
OnCollisionStay2D	Sent each frame where a collider on another object is touching this

	object's collider (2D physics only).
OnConnectedToServer	Called on the client when you have successfully connected to a server.
OnControllerColliderHit	OnControllerColliderHit is called when the controller hits a collider while performing a Move.
OnDestroy	Destroying the attached Behaviour will result in the game or Scene receiving OnDestroy.
OnDisconnectedFromServer	Called on the client when the connection was lost or you disconnected from the server.
OnDrawGizmos	Implement OnDrawGizmos if you want to draw gizmos that are also pickable and always drawn.
OnDrawGizmosSelected	Implement OnDrawGizmosSelected to draw a gizmo if the object is selected.
OnEnable	This function is called when the object becomes enabled and active.
OnFailedToConnect	Called on the client when a connection attempt fails for some reason.
OnFailedToConnectToMasterServer	Called on clients or servers when there is a problem connecting to the MasterServer.
OnGUI	OnGUI is called for rendering and handling GUI events.
OnJointBreak	Called when a joint attached to the same game object broke.
OnJointBreak2D	Called when a Joint2D attached to the same game object breaks.
OnMasterServerEvent	Called on clients or servers when reporting events from the MasterServer.
OnMouseDown	OnMouseDown is called when the user has pressed the mouse button while over the GUIElement or Collider.
OnMouseDrag	OnMouseDrag is called when the user has clicked on a GUIElement or Collider and is still holding down the mouse.
OnMouseEnter	Called when the mouse enters the GUIElement or Collider.
OnMouseExit	Called when the mouse is not any longer over the GUIElement or Collider.
OnMouseOver	Called every frame while the mouse is over the GUIElement or Collider.
OnMouseUp	OnMouseUp is called when the user has released the mouse button.
OnMouseUpAsButton	OnMouseUpAsButton is only called when the mouse is released over the same GUIElement or Collider as it was pressed.

OnNetworkInstantiate	Called on objects which have been network instantiated with Network.Instantiate.
OnParticleCollision	OnParticleCollision is called when a particle hits a Collider.
OnParticleSystemStopped	OnParticleSystemStopped is called when all particles in the system have died, and no new particles will be born. New particles cease to be created either after Stop is called, or when the duration property of a non-looping system has been exceeded.
OnParticleTrigger	OnParticleTrigger is called when any particles in a particle system meet the conditions in the trigger module.
OnPlayerConnected	Called on the server whenever a new player has successfully connected.
OnPlayerDisconnected	Called on the server whenever a player disconnected from the server.
OnPostRender	OnPostRender is called after a camera finished rendering the Scene.
OnPreCull	OnPreCull is called before a camera culls the Scene.
OnPreRender	OnPreRender is called before a camera starts rendering the Scene.
OnRenderImage	OnRenderImage is called after all rendering is complete to render image.
OnRenderObject	OnRenderObject is called after camera has rendered the Scene.
OnSerializeNetworkView	Used to customize synchronization of variables in a script watched by a network view.
OnServerInitialized	Called on the server whenever a Network.InitializeServer was invoked and has completed.
OnTransformChildrenChanged	This function is called when the list of children of the transform of the GameObject has changed.
OnTransformParentChanged	This function is called when the parent property of the transform of the GameObject has changed.
OnTriggerEnter	OnTriggerEnter is called when the Collider other enters the trigger.
OnTriggerEnter2D	Sent when another object enters a trigger collider attached to this object (2D physics only).
OnTriggerExit	OnTriggerExit is called when the Collider other has stopped touching the trigger.
OnTriggerExit2D	Sent when another object leaves a trigger collider attached to this object (2D physics only).
OnTriggerStay	OnTriggerStay is called once per physics update for every Collider other that is touching the trigger.

OnTriggerStay2D	Sent each frame where another object is within a trigger collider attached to this object (2D physics only).
OnValidate	This function is called when the script is loaded or a value is changed in the inspector (Called in the editor only).
OnWillRenderObject	OnWillRenderObject is called for each camera if the object is visible and not a UI element.
Reset	Reset to default values.
Start	Start is called on the frame when a script is enabled just before any of the Update methods are called the first time.
Update	Update is called every frame, if the MonoBehaviour is enabled.

Description

The size fit mode to use.

Properties

Unconstrained	Don't perform any resizing.
MinSize	Resize to the minimum size of the content.
PreferredSize	Resize to the preferred size of the content.

Description

Utility class for creating default implementations of builtin UI controls.

The recommended workflow for using UI controls with the UI system is to create a Prefab for each type of control and instantiate those when needed. This way changes can be made to the Prefabs which immediately have effect on all used instances.

However, in certain cases there can be reasons to create UI controls entirely from code. The DefaultControls class provide methods to create each of the builtin UI controls. The resulting objects are the same as are obtained from using the corresponding UI menu entries in the GameObject menu in the Editor.

An example use of this is creating menu items for custom new UI controls that mimics the ones that are builtin in Unity. Some such UI controls may contain other UI controls. For example, a scroll view contains scrollbars. By using the DefaultControls methods to create those parts, it is ensured that they are identical in look and setup to the ones provided in the menu items builtin with Unity.

Note that the details of the setup of the UI controls created by the methods in this class may change with later revisions of the UI system. As such, they are not guaranteed to be 100% backwards compatible. It is recommended not to rely on the specific hierarchies of the GameObjects created by these methods, and limit your code to only interface with the root GameObject created by each method.

Static Methods

CreateButton	Create a button.
CreateDropdown	Create a dropdown.
CreateImage	Create an image.
CreateInputField	Create an input field.
CreatePanel	Create a panel.
CreateRawImage	Create a raw image.
CreateScrollbar	Create a scrollbar.
CreateScrollView	Create a scroll view.
CreateSlider	Create a slider.
CreateText	Create a text object.
CreateToggle	Create a toggle.

Description

Object used to pass resources to use for the default controls.

Properties

background	Sprite used for background elements.
checkmark	Sprite used for representation of an "on" state when present, such as a checkmark.
dropdown	Sprite used to indicate that a button will open a dropdown when clicked.
inputField	Sprite used as background for input fields.
knob	Sprite used for knobs that can be dragged, such as on a slider.
mask	Sprite used for masking purposes, for example to be used for the viewport of a scroll view.
standard	The primary sprite to be used for graphical UI elements, used by the button, toggle, and dropdown controls, among others.

Implements interfaces:ICancelHandler, IEventSystemHandler, IPointerClickHandler, ISubmitHandler

Description

A standard dropdown that presents a list of options when clicked, of which one can be chosen.

The dropdown component is a Selectable. When an option is chosen, the label and/or image of the control changes to show the chosen option.

When a dropdown event occurs a callback is sent to any registered listeners of onValueChanged.

Properties

captionImage	The Image component to hold the image of the currently selected option.
captionText	The Text component to hold the text of the currently selected option.
itemImage	The Image component to hold the image of the item.
itemText	The Text component to hold the text of the item.
onValueChanged	A UnityEvent that is invoked when when a user has clicked one of the options in the dropdown list.
options	The list of possible options. A text string and an image can be specified for each option.
template	The Rect Transform of the template for the dropdown list.
value	The Value is the index number of the current selection in the Dropdown. 0 is the first option in the Dropdown, 1 is the second, and so on.

Public Methods

AddOptions	Add multiple options to the options of the Dropdown based on a list of OptionData objects.
ClearOptions	Clear the list of options in the Dropdown.
Hide	Hide the dropdown list.
OnCancel	Called by a BaseInputModule when a Cancel event occurs.
OnPointerClick	Handling for when the dropdown is 'clicked'.
OnSubmit	What to do when the event system sends a submit Event.
RefreshShownValue	Refreshes the text and image (if available) of the currently selected option.If you have modified the list of options, you should call this method afterwards to ensure that the

	visual state of the dropdown corresponds to the updated options.
Show	Show the dropdown list.

Protected Methods

CreateBlocker	Override this method to implement a different way to obtain a blocker GameObject.
CreateDropdownList	Override this method to implement a different way to obtain a dropdown list GameObject.
CreateItem	Override this method to implement a different way to obtain an option item.
DestroyBlocker	Override this method to implement a different way to dispose of a blocker GameObject that blocks clicks to other controls while the dropdown list is open.
DestroyDropdownList	Override this method to implement a different way to dispose of a dropdown list GameObject.
DestroyItem	Override this method to implement a different way to dispose of an option item.

Inherited Members

Static Properties

allSelectables	List of all the selectable objects currently active in the Scene.

Properties

enabled	Enabled Behaviours are Updated, disabled Behaviours are not.
isActiveAndEnabled	Has the Behaviour had active and enabled called?
gameObject	The game object this component is attached to. A component is always attached to a game object.
tag	The tag of this game object.
transform	The Transform attached to this GameObject.
runInEditMode	Allow a specific instance of a MonoBehaviour to run in edit mode (only available in the editor).
useGUILayout	Disabling this lets you skip the GUI layout phase.
hideFlags	Should the object be hidden, saved with the Scene or modifiable by the user?
name	The name of the object.